LEARN PYTHON PROGRAMMING

this book will teach you about the language,
data analysis and algorithms and will level up your skills
in computer programming to become an expert Pythonista

MICHAEL LEARN

LEARN PYTHON PROGRAMMING

this book will teach you about the language,
data analysis and algorithms and will level up your skills
in computer programming to become an expert Pythonista

MICHAEL LEARN

Learn python programming

---

this book will teach you about the language, data analysis and algorithms and will level up your skills in computer programming to become an expert Pythonista

[Michae Learn]

Text Copyright © [Michael Learn]
All rights reserved. No part of this guide may be reproduced in any form without permission in writing from the publisher except in the case of brief quotations embodied in critical articles or reviews.

Legal & Disclaimer

The information contained in this book and its contents is not designed to replace or take the place of any form of medical or professional advice; and is not meant to replace the need for independent medical, financial, legal or other professional advice or services, as may be required. The content and information in this book has been provided for educational and entertainment purposes only.

The content and information contained in this book has been compiled from sources deemed reliable, and it is accurate to the best of the Author's knowledge, information and belief. However, the Author cannot guarantee its accuracy and validity and cannot be held liable for any errors and/or omissions. Further, changes are periodically made to this book as and when needed. Where appropriate and/or necessary, you must consult a professional (including but not limited to your doctor, attorney, financial advisor or such other professional advisor) before using any of the suggested remedies, techniques, or information in this book.

Upon using the contents and information contained in this book, you agree to hold harmless the Author from and against any damages, costs, and expenses, including any legal fees potentially resulting from the application of any of the information provided by this book. This disclaimer applies to any loss, damages or injury caused by the use and application, whether directly or indirectly, of any advice or information presented, whether for breach of contract, tort, negligence, personal injury, criminal intent, or under any other cause of action.

You agree to accept all risks of using the information presented inside this book.

You agree that by continuing to read this book, where appropriate and/or necessary, you shall consult a professional (including but not limited to your doctor, attorney, or financial advisor or such other advisor as needed) before using any of the suggested remedies, techniques, or information in this book.

Table of Contents

Introduction

- A Brief Application of Python
- Why you should select Python as your Language
  - Reasons you should consider to write Software Applications using Python

Chapter 1. what is python programing

Chapter 2. How to install python

- How to Install Python in Mac OS X
- Install and Run Python in Linux
- Install Python in Windows
- Python Interaction

Chapter 3. Data types used in python

Chapter 4. methods and functions of python

- Conditional programming
- Loops

Chapter 5. :processing data analysis and algorithms

Chapter 6. Object oriented programming in python

- Classes and Objects

Chapter 7. Teaching you the python language

- Why python
- How to Start Learning Python
  - Step 1: Download and Install
  - Step 2: Instructional Exercise
- The Basics of Working with Python
  - Python Keywords

Naming Identifiers in Your Code

Python Comments

Python Statements

Python as an Object-Oriented Language

Chapter 8. Important python frameworks

Registering Python Programs

Chapter 9. Introduction to data analysis

Chapter 10. Algorithms for starters

Arithmetic Operators

Comparison Operators

Equality Comparison-Floating-Point Values

Logical Operators

Logical Expressions including Boolean Operands

Computation of Non-Boolean Values in a Boolean Context

Chapter 11. Conclusion

Introduction

Why was Python developed?

The reason why Guido Van Rossum embarked on the move to design a new programming language is that he wanted a language that could offer a simple syntax just like the ABC. This motivation led to the development of a new language named Python.

But you may be wondering why just the name Python**?**

First, this language wasn't named after the huge snake called python. No! One of the interests of Rossum was watching comedy. He was a great fan of the comedy series in the late seventies. As a result, the name of the language was borrowed from the "Monty Python's Flying Circus."

Properties of Python

- **Easy to learn** – The syntax of Python is simple and beautiful. Additionally, Python programmers enjoy writing its syntax than other languages. Python simplifies the art of programming and allows the developer to concentrate on the solution instead of the syntax. For a newbie, this is a great choice to start your Python career.
- **Portability** – When it comes to Python portability, it offers you the ability to run Python on different platforms without making any changes.
- **Python is described as a high-level language** – In other words, you don't need to be scared of tedious tasks such as memory management and so on. Alternatively, whenever you execute a Python code, it will automatically change the language to a language that your computer understands. No need to be worried about any lower-level operations.
- **Object-oriented** – Since it is an object-oriented language, it will allow you to compute solutions for the most difficult problems. Object-Oriented Programming makes it possible to divide a large problem into smaller parts by building objects.
- **Has a huge standard library to compute common tasks** – Python has different standard libraries for the programmer to use. As a result, you will not write all the lines of code

yourself. Instead, you will only import the library of the relevant code.

Python is a multi-purpose language created by Guido van Rossum. The language boasts of a simple syntax that makes it easy for a new learner to understand and use. This chapter will introduce the basics of the Python language. Stay tuned.

Python is described as a general-purpose language. It has many applications and therefore, you can use it to accomplish many different functions.

The syntax of a python language is clean and the length of the code is short. Developers who have used Python at one point of their lives will express how fun it was to code with Python. Python offers you a chance to think more about the task at hand instead of the language syntax.

Some history of Python

The design of the Python language started back in the 1980s and it was first launched in February 1991.

A Brief Application of Python

Web Applications

You develop a scalable Web application using CMS and frameworks that are created on Python. Popular environments for developing web applications include Pyramid, Django, Django CMS, and Phone.

Other popular websites like Instagram, Mozilla, and Reddit are written in Python language.

Scientific and Numeric Calculations

There are different Python libraries designed for Scientific and numeric calculations. Libraries such as Numpy and Scipy use Python for general computing purpose. And, there are specially designed libraries like AstroPy for Astronomy, and so on.

Additionally, the Python language is highly applied in data mining, machine

learning, and deep learning.

A great Language for Tech Programmers

The Python language is an important tool used to demonstrate programming to newbies and children. It is a great language that has important capabilities and features. However, it is one of the easiest languages to learn because it has a simple syntax.

Building Software Prototypes

Compared to Java and C++, Python is a bit slow. It may not be a great choice when resources are restricted and efficiency is made compulsory.

But Python is a powerful language to build prototypes. For instance: You can apply the Pygame library to develop the prototype of your game first. If you enjoy the prototype, you can decide to use C++ to develop the actual game.

Why you should select Python as your Language

Simple, Beautiful Syntax

Let no one deceive you; it is fun to program in Python. It is easier to understand a line of code and write a Python code. The syntax looks natural. For example:

```
b = 4

c = 3

sum = b + c

Print (sum)
```

Even if it is your first time to program, you can manage to guess the function of this simple program.

Not a Very Strict Language

No need to define the type of variable in Python language. More so, you don’t need to end a line with a semicolon. However, Python demands that you adhere to good practices like the right indentation. These little things simplify learning Python for beginners.

The Language is Expressive

Python will let you write powerful programs that have many functions using fewer lines of code.

Lovely Community and Support

There is a large community of Python developers. You will come across different active online forums, which can be helpful while learning how to program in Python.

Reasons you should consider to write Software Applications using Python

Easy to Read and Maintain

While creating an application, you need to concentrate on the brilliant nature of the source code to make the process of updating and maintaining the code easy. The Python Syntax permits you to demonstrate concepts without writing extra code.

Besides this, Python provides the chance to use English terms instead of punctuations. For that reason, one can use Python to build custom applications without writing extra code. When you write code that is easy to read, it simplifies the process of updating the software.

Compatible with Key Platforms and Systems

As said before, Python can run in different operating systems. You can even decide to use Python Interprets to run the code on a defined platform. Additionally, Python is an interpreted programming language. In other words, you can run the same piece of code on different platforms without recompiling. Therefore, you don’t need to recompile the code once you make any changes. In fact, you can run the altered code without recompiling and confirm the effect of changes done to the code instantly. This feature allows developers to change the code without increasing the building time.

Many Open Source Technologies and Frameworks

Since Python is an open source language, it helps save the cost of the software. Why? You can include different open source Python frameworks, development tools, and libraries to reduce the development time without increasing the cost of development.

Breakdown the Intricate Software Development

You can use Python to build both web and desktop applications. Still, you can use it to develop complex scientific programs. Python is designed with properties to support data analysis and visualization. You can use the data analysis properties of Python to build big custom solutions without spending

extra energy and time. Alternatively, the data libraries and visualizations provided by Python language allow you to visualize and present data in a friendly and effective way.

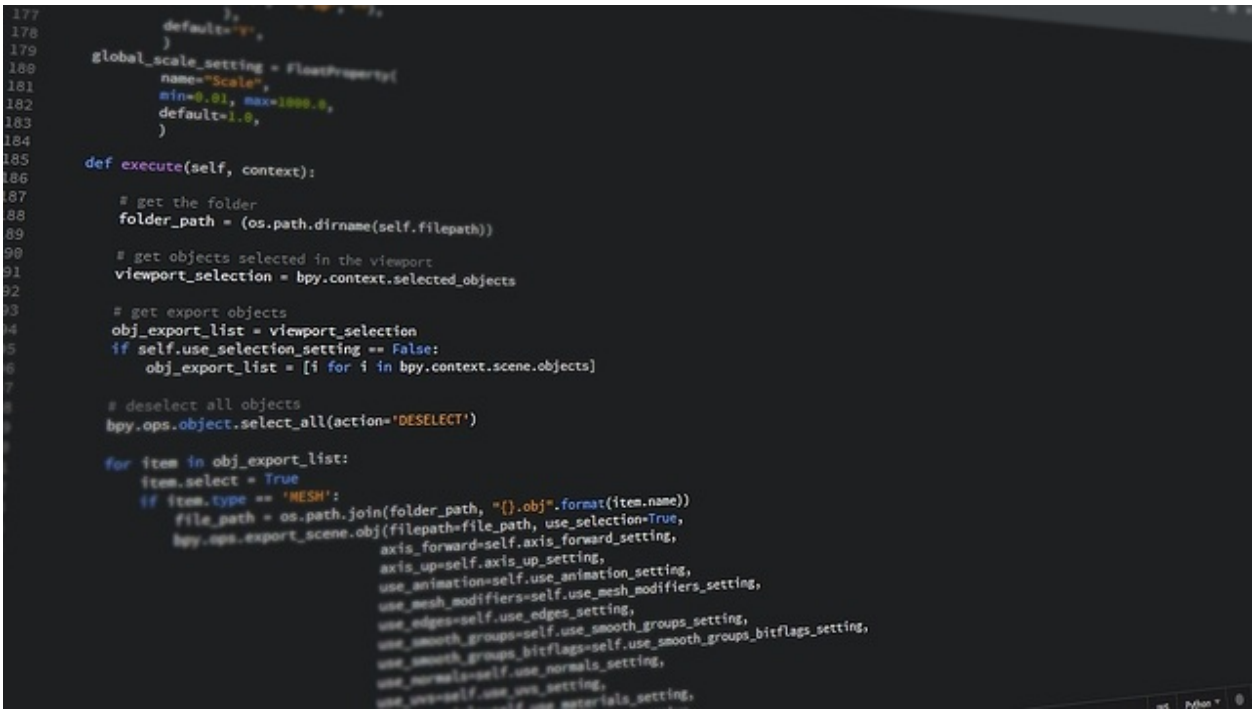
Apply Test Driven Development (TDD)

With Python, it is possible to develop a software prototype application very fast. Furthermore, the software application can be developed directly from the prototype by refactoring the python code.

Still, Python simplifies coding and testing through the use of TDD methodology. You can quickly write the relevant tests before writing code and adopting the tests to examine the application code. Also, the tests can help confirm whether the application fulfills predefined requirements depending on the source code.

However, Python has its own drawbacks. For example, it doesn’t have built-in features available in other modern programming languages. As a result, you just need to use Python libraries and frameworks to increase custom software development. Also, Python is slower than other languages like Java and C++. This means you will need to boost the Python application by implementing changes to the application code. But still, you can use Python to simplify software maintenance and speed up the process of software development.

Chapter 1. what is python programing



Learning a new programming language can be a great experience. You may want to learn it so you can know more things about your computer and understand how it works, how a website works, and perhaps you want to learn so you can create your own applications and programs. But if you have never worked with your computer in this way or with coding in the past, then it is hard to figure out how to get started. With a little search, you will notice that there are many coding languages available that you can work with besides Python. From Java to C++ and so much more, it’s hard to determine which coding language is the best for you.

As a beginner and someone who has never done anything with coding in the past, Python is one of the best coding languages to work with. The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

Python has long been considered a beginner's language because it is so easy to learn and you will be able to understand it right from the beginning. This is just one of the many reasons why you should choose to work with Python. You will also like there is a large active community devoted to this programming language and what’s good is that it’s open source so you can get started without

literally having to pay anything. This language will also work on any operating system so it won't matter which computer you want to use the language on.

If you're browsing the introductory pages of this book on a preview or something of the like, on the fence as to whether or not you want to learn Python, well, I can tell you that you probably should.

Well, Python is basically an incredibly useful and powerful language. It's present essentially everywhere. Everything from the scripting side of video games (or the video games themselves) to intensive server-side web applications to the plethora of deep and responsive desktop applications that have been built with it.

When should you use Python? The answer depends upon exactly what you're going to do. But since you're a beginner, I say you should learn anyway.

As you continue to grow as a programmer after this book, you're going to learn when you should and shouldn't use Python just as a matter of intuition. Python is an absolutely fantastic language, but the place where it fails is when you have to get extremely close to a computer's hardware or write incredibly efficient programs. In these areas, Python doesn't excel.

However, that's not to say it doesn't have its perks. In fact, I'd say that's one of the few places that Python falls flat. And what it lacks there, it makes up for in other areas. For example, development time in Python is generally extremely low in comparison to other languages. This is super easy to illustrate. Compare the following excerpts of code, the first from C++, the second from Java, and the last from Python.

Despite being the coding language that beginners like to work with, this doesn't mean that there aren't a lot of advantages that come with working with this language. The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

Python on its own is capable of writing great codes in the process, and you can also combine it with a few other languages so that you can create as many strings of code that you like. Now, let's take some time to look at Python and all the things that you need to know to get started on using this coding language.

Why should I learn Python?

As we mentioned before, there are actually a lot of different coding languages that you can learn about and use. So, if there are a lot of choices out there why would you want to go with the Python coding language in the first place? Many people, both experts, and beginners all choose to go with Python because it is easy to learn, easy to read, and it is capable of creating large, challenging codes that you might want to write. These are the reasons to work with coding:

Simple enough to read

You will find that Python is a programming language that is very easy to read, even if you are a beginner. In fact, when it is compared to some of the other coding languages, it is one of the most readable languages. Since this is an easy coding language to go with, many beginners like that they can catch on so quickly and that they will understand what they are doing in no time.

Free

Another benefit of going with Python is that it is free to use. There are some computer coding languages that you would have to pay for so you can use them and this can be quite expensive especially if you want to learn how to use more than one of them. Python is free to use, so you don't have to worry about this problem.

Fast

Even though this language is easy enough for any beginner to learn, Python is still considered one of the high-level languages that you can learn about. This means that when you make a program and generate your own codes by using Python, you will see that the execution is nice and quick. There are some coding languages that are harder to work with or can't go as fast as you want them to, but this is a problem you won't have when using Python.

Works on a variety of platforms

You can work with the Python language no matter which platform you would like to use it on. Linux is the operating system that a lot of people will choose to go with but you can still work with Python even if you are on a Windows or Mac computer. This can actually use Python without having to make any major changes to your current setup.

A big library to work with

Once you start to get familiar with Python, you will notice that it comes with a large library. This is good news for beginners because the library is what

contains all the functions, codes and other things that you need to make the language work for you. This library will help make sure that you can do some useful things when trying to make your own code.

A large community

Whether you have worked with coding language in the past or not, it is nice to know that there is a large community of Python users that will help you out if you ever get stuck. Any time that you need some ideas like a new project or if you just have a question, or if you want to learn something new, there is a library of information to provide you with the information that you need to help you get started.

There are a ton of people, both beginners, and experts, who use the Python language over other programming languages because unlike those, it is simple and easy to work with. Whether or not you have worked with a coding language in the past, you will find that Python may just be the answer that you've been looking for.

Download Python

You are ready to get started with some of your own coding with Python. You want to get the program downloaded so you can learn how to make some of your own amazing codes. But before you can do this, you need to make sure that all the right components are downloaded on your computer. The good news is that since Python is compatible with all the operating systems currently in use, you just need to start by turning on your personal computer and then go to the next step.

Since this coding language is open sourced and completely free, all you have to do is to visit the Python website, at **www.python.com** , to get started. There are a few different versions to go with, so you should pick out the one that you think you can use. There are versions available for each operating system so make sure to look at this as well before choosing the version that you want to work with. Once you decide on which version would work best for your computer, you just need to click on the prompts that show up and do as instructed to get it downloaded on your computer.

While you are waiting for the Python program to finish downloading, there are also a few other tools that you will need that can help with writing these codes. This text editor is the tool which you will use to write out the codes that you want to make. You will also need an IDE or an 'Integrated Development

Environment’ which you can use to read and execute the codes that you want.

Let’s start with the text editor. It is not necessary that you go with a program that is too complicated to make this work. The notepad text editor found on a Windows computer or some other program will work just fine. There are a lot of options that you can choose from here, but Notepad or something similar is free, so it is a good option for you to work with.

Then you also need to take some time to work with the IDE. If you don't add this to your computer, it will be impossible to write out or execute the code you want to work on. There are a few different options when it comes to the IDE, so do some shopping around to see which kind will offer all the features that you are looking for. There is one IDE that is available, and you can get it through Python so that you can keep things simple, but there are also other options available if you need more choices.

Once you have downloaded the Python program as well as the IDE and the text editor that you want to work with it is time to work on writing your own code. The process is simple, but we do need to look at a few things first and learn some of the basics so that you can start to write your own Python code in no time.

The basics you should know about Python

As you get started with writing your own codes in this language, you will start to see that there are many parts that come with them. Being able to understand how these basic parts work will help you to write the codes that are needed in your program. This section will quickly go over some of the basics that you need to know before you start working with Python.

The keywords

First, we need to take a look at these important keywords in the Python language. Similar to what you will find in other coding languages there is a list of keywords in Python that are meant to tell your text editor what to do. These keywords are reserved, and you should only use them for their intended purposes if you want to avoid any issues with your code writing. They are basically the commands that will tell your compiler how to behave, and they have to remain in reserve so that you can execute the code without unnecessary issues complicating the process.

How to name the identifiers

Once you are ready to write your own codes in Python, there are a few parts of the code you will encounter, and they are called ‘identifiers.’ These ‘identifiers’ go by a lot of different names so it can get a bit confusing, but they will help by getting your code to run the way you want it. You will notice that some of the names that we used throughout this guidebook for identifiers include functions, classes, variables, and entities. There are some rules to follow when naming your identifier but these rules will be the same no matter what type of identifier you are currently working with. Some of the rules that you should know about include:

Naming these identifiers is simple. You can use both the lower-case and the upper-case letters any time that you would like. Numbers and the underscore symbol are allowed as well. Any combination of the above are allowed, just make sure that you aren’t adding in spaces in this name. So, you would name it ‘MyPythonProgram’ rather than My Python Program.

The identifiers can't start with any numbers. It is fine to use numbers throughout the name, but you should never have this at the beginning of the name or in the first character at least. This will result in an error when you type it into the compiler so if you are uncertain about why you see an error, be sure to double-check if this is the reason.

The identifier shouldn’t have one of the keywords that we talked about before. Doing this will confuse your compiler so just avoid it completely.

As you can see these rules are not too complicated to keep in mind and outside of them, you can name your identifiers whatever you would like. Now, if you are working on the code and you forget one of the rules and give your identifier a name that is not allowed then the compiler will provide you with a syntax error which indicates that you need to go and fix it.

Statements

You will use a lot of ‘statements’ when writing out a code in Python. These are useful for your code because they are just strings of code, these include some of the other parts we have talked about, that you will send over to the compiler to execute. You can write out pretty much any type of statement that you would like in your code, as long as it is written out in a way that your compiler is going to understand. You can choose to make a statement short, such as having a few lines with it, or make it longer, there really aren't many rules that dictate how you should do it.

Comments

The next component that you will use quite a bit in making your own code is the ‘comments.’ These comments are helpful because if someone else is taking a look through the code that you wrote and they need some clarification on what exactly it is you are talking about. You can write out the comments so that others who want to read through or use the code will know what is going on. They are basically little notes inside the code that help you and other programmers to know what is happening during the application.

It is pretty easy to write out the comments that you want to add to your code. You simply need to use the (#) sign in front of the comment that you want to write. So, you could write out something like #this is my Python code. When the comment is done, you just hit return and start writing the rest of the code on the next line. As long as you have that sign right in front of the comment, then your compiler can just skip over it and will not read out what you put in the comment.

You have the freedom to choose how many comments you would like to leave inside of your code. Sometimes the code will just need a few of them while other times the code may need a lot of comments. Keep the comments down to the ones that you really need and do not waste time or space writing out more comments than needed.

Functions

‘Functions’ are another important part of the Python programming language. This is basically a part of the code that you can reuse, and it is often used to finish off one action in the code. Functions are great because they are more effective than other options, which means you can work on your code without wasting a lot of time. There are a lot of functions that are found in the Python language so you can benefit from this, plus, you are able to write out some of your own functions as well.

While you are working on your code, you have to make sure that you are defining the function. Once you are able to define the function and it is considered finalized, it is time to execute it to work in the code. You have the choice to call it up from the Python prompt, or you can call it up from another function.

When this is placed into your compiler, you will see the two statements that we wrote out inside of the code that will come up just like a message. You would

call up a function, and you can always change up the statements that are inside the code to figure out how you want to execute them later.

Working with variables

And finally, it is important that you spend some time to focus on variables. Variables are important because they will save up spots on your computer's memory to hold onto parts of your code. When creating a new variable, in effect, you are making sure that some of the space on your computer will be reserved. In some cases, such as when you are working with data types, your interpreter will do the work of deciding where this information should be stored and this helps speed up the process.

When it comes to working with variables, your job is to simply make sure that the right variables are lining up with the right values. This will ensure that the right parts show up in your code at the right time. The good news is that you are able to give the variable any value that you would like, but do check if it actually works inside of your code first. When you are ready to assign a new value to a variable, you can use the equal sign to make this happen.

Chapter 2. How to install python

Guide to install Python on your Windows computer, Linux, and Mac OS X.

How to Install Python in Mac OS X

1. Navigate to the Python Download page on the official website and click “Download Python” (There will be a version name at the end.)
2. Once the download completes, open the package and begin to follow the instructions. You will then see the notification “The installation was successful” when Python completes the installation successfully.
3. It is advised to download a great text editor before you can start the installation process. For new beginners, you should download the Sublime Text. It is available for free.
4. The installation process of Sublime text is quite simple.
5. Write the code and save it. For newbies, here’s a code to use:

Print (“Hello, World!”)

The simple program prints, “Hello, World!”

7. Navigate to Tool > Build. Here, you can move to the bottom to see the results of your program.

Install and Run Python in Linux

1. First, you need to install the dependencies below.

```
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev
libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
```

2. Navigate to the Download Python page located on the official site and click “Download Python 3.6.0”. If you see a different version, don’t hesitate to download it.
3. Go to the terminal and locate the path where the file is downloaded and execute the following command:

$ tar -xvf Python-3.6.0.tgz

This line will remove your zipped file. Just know that the filename will be different if you downloaded a different version. So you should ensure you use the right filename.

4. Now shift to the extracted directory.

```
$ cd Python-3.6.0
```

5. Use these commands to compile Python source code on your OS.

```
$ ./configure
$ make
$ make install
```

6. It is advised that you install Sublime Text if you are a new beginner. If you are using Ubuntu, then you should run the following commands to install Sublime Text:

```
$ sudo add-apt-repository -y ppa:webupd8team/sublime-text-2
$ sudo apt-get update
$ sudo apt-get install sublime-text
```

7. Now open Sublime text. To build a new file, navigate to File>New File
8. Save the file using the .py file extension such as beginner.py
9. Write the code and save it. For beginners, you can use the following code:

Print ("Where are you?")

This program will display, "Where are you?"

10. Navigate to the Tool > Build. You can view the output at the bottom of the Sublime Text.

Install Python in Windows

1. Navigate to the Download Python page found on the official site and click on the Download Python 3.6.0.
2. Once the download is complete, open the file and start the installation process. Once the installation of Python is done, IDLE is installed alongside.
3. Open the IDLE, paste the code below and press enter.

Print (“Hello, World!”)

4. To develop a file in IDLE, navigate to the File > New Window.
5. Write a Python code and save it with a .py file extension.
6. Navigate to the Run > Run module and view the output.

Python Interaction

As said in the first chapter, the Python language is popular because of the level of flexibility and dynamic that a developer can use to achieve different things. For example, you can use Python interactively when you want to test a statement on a line-by-line basis or when you are trying to learn new properties. You can also use it in script mode when you have a whole file of statements that you want to interpret.

Python has the Command Line Window or the IDLE that allow programmers to use it interactively.

Let’s look at the command line interaction

The command line is the simplest way for one to work with Python. With this tool, one can visualize the way Python works as it responds to every finished typed command. It might not be the best tool to use to interact with Python but still is the easiest means to use to learn how Python works.

How to start Python

There are different methods that you can use to access the command line of Python depending on the type of operating system running in your computer:

- **Also, users of Windows can navigate to the folder that has a shortcut and click on the Python command line.**
- **For those using Linux and Mac OS systems, you need to run the Terminal Tool and type the Python command to start a session.**

Computer commands are used to instruct the machine on what to do. Therefore, if you want Python to perform a given task, you have to type commands that it will interpret. Next, Python will translate the commands to instructions which the machine is familiar with and execute.

To understand how Python works, you will use the print command to print a simple message: Welcome to Python.

1. **First, open your Python command line tool.**
2. **Then type the following code at the >>> prompt**

Print ("Welcome to Python!")

3. **Now press enter to instruct Python that you are done writing the command.**
4. **Next, you will see the message Welcome to Python! On the command line window.**

In the above example, Python responded the right way because you typed the correct command in the right format. Let's say you typed the command incorrectly; then Python will have responded with the following message: Syntax error: Invalid syntax.

When you come across a syntax error message, then know that you have entered an incomplete statement in your code. For example, typing the keyword print in the capital letter will result in a syntax error.

When you are programming with Python, it gives you room to do away with the print command completely by only typing your statements within quotes. For example: "Welcome to Python!"

How to exit from Python

Once you are done using Python, you can exit by typing the following commands:

Quit ()

Exit ()

Control-Z then press Enter

IDLE: Python's Integrated Development Environment

When you first open the IDLE, it will start in the Shell. The shell is an interactive window for you to type in Python code and monitor the output in the same window.

Note: On windows, when you click a Python file on the desktop, the Windows

system will run the program. However, if you right click on the file, there would be an option for the Edit with IDLE. To modify an existing Python file, open the IDLE, and then open the file via the File menu.

First Python Program

Up until now, you have already written the first program in Python. The “Hello World Program” is always the first program to write in any programming language.

In the following example, we shall look at a different type of program. To start, open the IDLE and create a new window. If you don’t know how to create a new window, just click on “New Window located under the File Menu”. Then you can type the following program:

```
temp = eval(input('Enter a temperature in Celsius: '))
print('In Fahrenheit, that is', 9/5*temp+32)
```

Navigate to the Run menu of the IDLE and select the Run Module. If you don’t want to do so, just press F5. The IDLE will request you to save the file and you should proceed and so. Make sure you append .py to the filename because the IDLE will not automatically do so. The .py extension will tell the IDLE the file type is for Python and thus include colors to make the program easier to read.

Once you save the program, it will be executed in the shell window. The program will request you for a temperature. Type the value 20 and click Enter. The output of the program will be like this:

```
Enter a temperature in Celsius: 20
In Fahrenheit, that is 68.0
```

Let’s see how the program works. The first line requests the user to type the temperature. The role of the input function is to prompt the user to enter something and to store what the user types. The part enclosed in quotes is what the user sees. This is referred to as a “string” and it will appear to the user of the program exactly as it is in the code itself. The “eval” is a function but you won’t understand until when we shall look at python functions. So for now, just know that it is used when we want to get numerical input.

The next program you will learn is one that calculates the average of two

numbers entered by the user.

```
num1 = eval(input('Enter the first number: '))
num2 = eval(input('Enter the second number: '))
print('The average of the numbers you entered is', (num1+num2)/2)
```

Python language has a great feature known as line indentation. Python applies the following feature on things that you will learn soon. On the flipside, spaces in other places aren't really important. For example, these lines have the same effect:

```
print('Hello world!')
print ('Hello world!')
print( 'Hello world!' )
```

In general, computers will only run depending on the command you type and they always interpret things literally.

On the other hand, Python operates based on the placement of commas and parentheses so that it can know what's what. It is not a great language to predict whatever you mean so you will need to be accurate. At first, it will sound frustrating to get all commas and parentheses in the correct places but after some time, it will be more natural. Even when you program for a long time, you will still forget something. Luckily, the Python interpreter helps show you areas where you have made mistakes.

Accepting input

The input function is an easy way for your program to receive information from users through the program. For example:

```
name = input('Enter your name: ')
print('Hello, ', name)
```

The standard syntax is:

Variable name = input (message to user)

This structure is used when you want to get a text from a user. However, if you

want to get numbers from the user to include in computations, you will have to go an extra mile. Check out this example:

```
num = eval(input('Enter a number: '))
print('Your number squared:', num*num)
```

The “eval” function will change the text typed by the user into a number. One great feature of this is that you can type expressions such as 2*4+ 9 and the “eval” will determine the result for you.

Note: When you run a program and nothing seems to be taking place, press enter. Sometimes, there is always a glitch in IDLE that frequently takes place with input statements.

Printing

Consider the following example:

Print (‘Hello there’)

The function “print” should be enclosed with parenthesis around its arguments. In the following program, the argument is the string ‘Hello there.’ Anything that is enclosed in the quotes will be displayed exactly as it appears. In the first program shown below, it will print ‘4+ 2’ while the second program will print 8.

print (‘4+2’)

print (4+ 4)

To display different things at once, you need to use commas to separate the items. However, Python will automatically insert spaces between the items. For example:

```
print('The value of 3+4 is', 3+4)
print('A', 1, 'XYZ', 2)
```

Output

```
The value of 3+4 is 7
A 1 XYZ 2
```

Chapter 3. Data types used in python

Programming involves working with data of different types. Python programs make use of objects which represent different data types. Objects may be built-in, defined by a programmer, or imported from libraries or modules.

Python provides the following common data types:

numbers

strings

Boolean values

lists

tuples

dictionaries

Numbers

There are 3 numeric data types in Python 3:

- **integers**

- **floating point numbers**
- **complex numbers**

Integers (int)

Integers are whole numbers with no decimal point or fractional part. This numeric type can have unlimited size in Python 3 and includes zero, positive, or negative numbers. Python supports normal integers, binary, octal, and hexadecimal literals.

Normal integers

Examples:

0, 25, -50, 56732164987123

Binary literal (base 2)

You can identify a binary literal by the form 0 (zero) followed by either an uppercase B or lowercase b.

Examples:

```
>>> 0b1010

10

>>> bin_lit = 0b1010

>>> print(bin_lit)

10
```

Octal literal (base 8)

An octal literal is a number prefixed with zero and uppercase O or lowercase o.

Examples:

```
>>> 0O24

20

>>> 0o24

20

>>> oct_lit = 0O24

>>> print(oct_lit)

20
```

Hexadecimal literal (base 16)

A hexadecimal literal is prefaced by a zero and an uppercase X or lowercase x.

Examples:

```
>>> 0XA0F

2575

>>> 0xa0f

2575

>>> hex_litreral = 0xa0f

>>>print(hex_literal)

2575
```

Converting Integers to String

The functions oct(), bin(), and hex() can be used to convert an integer to a desired string equivalent.

For example, here is how you can convert the integer 50 to its octal, binary, and hexadecimal literals:

integer to octal literal:

```
>>> oct(50)

'0o62'
```

integer to binary literal:

```
>>> bin(50)

'0b110010'
```

integer to hexadecimal literal

```
>>> hex(50)

'0x32'
```

Floating-Point Numbers (Floats)

Floats are real numbers with a decimal point and fractional part.

Examples:

10.75, 987345764.24 , 1.0, 18.98729750582

Complex Numbers

A complex number is a pair of a real and an imaginary number. It is written in the format ‘a + bj’ or ‘a + bJ where ‘a’ and ‘b’ are floats and J or j is an imaginary number. The ‘J’ or ‘j’ stands for the square root of -1, an imaginary number.

For example:

```
>>> x = 4 + 5j

>>> y = 2 - 2j

>>> xy = x + y

>>> print(xy)

(6+3j)
```

While Python can intuitively handle the conversion of mixed number types to a common type, syntax requirements may require the explicit conversion of numeric types. The use of the right keyword and parameter will compel Python to implement the conversion.

Integer to a float:

syntax: float(x)

Example:

```
>>> float(55)

55.0
```

Float to an integer:

syntax: int(x)

```
>>> int(95.5)

95
```

Integer to a complex number:

syntax: complex(x)

```
>>> complex(50)

(50+0j)
```

Float to a complex number:

syntax: complex(x)

```
>>> complex(20.5)
(20.5+0j)
```

Numeric pair expression to a complex number:

syntax: complex(x, y)

```
>>> complex (4, 2)
(4 + 2j)
```

String

A string is an ordered series of Unicode characters which may consist of letters, numbers, special symbols, or a combination of these types. A string is an immutable data type which only means that once it is created by assignment, you will no longer be able to modify it.

A string is defined in an assignment statement where it is enclosed in a pair of single or double quotes.

Examples:

```
>>> string_1 = 'A string is immutable.'
>>> string_2 = "A string is a sequence."
```

If a string enclosed in a single quote uses a single quite within it, you must escape the single quote by placing a backslash before it.

For example:

```
>>> string_negative = 'I don\'t have any idea.'
```

Here is what the screen will display if you print ‘string_negative’:

```
>>> print(string_negative)
I don't have any idea.
```

Similarly, you will need to escape a double quote within a string with a backslash when the entire string is enclosed in double quotes.

For example:

```
>>> double_quote = "She reminded everyone: \"You are supposed to use a backslash to escape a double quote inside double quotes.\""
```

This would be your output when you print ‘double_quote’:

```
>>> print(double_quote)
```

She reminded everyone: "You are supposed to use a backslash to escape a double quote inside double quotes."

The same principle will apply when you have to use a backslash as a character within the string.

Accessing String Value

A string can be accessed by using the square brackets or index operator []. The initial character or substring takes zero as it index and the rest are numbered sequentially.

To illustrate string indexing, first define a variable ‘new_string’ by assigning the string “Programmer’s Handbook”.

```
>>> new_string = "Programmer's Handbook"
```

To access the first character on new_string:

```
>>> new_string[0]
'P'
```

To access the 7**th** character on new_string, enclose 6 on the square brackets:

```
>>> new_string[6]
'm'
```

To access the substring on index 12, a blank:

```
>>> new_string[12]
' '
```

To access the last character in the string, you may use the len() function or negative indexing.

The len() function is used to determine how many characters are in a string. Once you have this information, you can simply deduct 1 to get the index of the last character in a string.

The syntax is: string[len(string)-1]

For example:

```
>>> new_string[len(new_string)-1]
'k'
```

An easier way to access the last character is by using negative indexing where the last character has the index -1, the penultimate character -2, and so on.

```
>>> new_string[-1]
'k'
>>> new_string[-4]
'b'
```

Getting the Length of a String

The length or size of a string refers to the number of characters that comprise it. The built-in function len() is used to determine the length of a string.

Example:

```
>>> string_a = 'Programming'
>>> len(string_a)
11
```

Slicing Strings

The slicing operator [:] is used to create substrings from a given string. The colon inside the square brackets is used to separate two indices from each other. The first number indicates the index of what would be the initial character of the substring. The second number refers to the first character to be dropped from the original string.

For example, here's the original string:

```
>>> string_1 = "Philosophy"
```

To display a new substring 'sop', supply the index 5 and 8 inside the square brackets:

```
>>> string_1[5:8]
'sop'
```

If you intend to start the substring with the same character as the original string, you can drop the first index. The same rule applies when you intend to end the substring on the same character as the original string.

For example:

```
>>> string_1[0:5]
```

'Philo'

>>> string_1[5:]

'sophy'

Take note that since a string is immutable, any manipulation done to it will not change the string stored inside the variable. For instance, if you print string_1 at this point, Python will display the original value.

>>> print(string_1)

Philosophy

Concatenating Strings

The + operator can be used to combine several string into a single string.

To illustrate:

>>> string_1 = "Stephen King"

>>> string_2 = "author"

>>> string_3 = "supernatural fiction"

>>> string_4 = "fantasy"

>>> string_1 + " is an " + string_2 + " of " + string_3 + " and " + string_4 + " books."

'Stephen King is an author of supernatural fiction and fantasy books.'

Repeating a String

A string or a concatenation of strings may be repeated by using the exponential operator * and a number to represent the number of times that the string will be repeated.

For example, to repeat the string **<><>** three times, you will need to type it on the >>> prompt followed by *3:

>>> "**<><>**" *3

Here's the output:

'**<><>****<><>****<><>**'

You can store the same string in a variable and apply the exponential operator on the variable:

```
>>> my_string = "**<><>**"
>>> my_string *3
'**<><>****<><>****<><>**'
```

The upper() and lower() functions

The upper() and lower() functions are used to display the entire string in uppercase or lowercase letters.

For example, create the variable ‘animal’ by assigning the string ‘elephant’:

```
>>> animal = "elephant"
```

To display the entire string in uppercase letters, type:

```
>>>print(animal.upper())
```

The screen will display the output:

ELEPHANT

If you want the display the entire string in lowercase letter, type:

```
>>>print(animal.lower())
```

The output would be:

elephant

The use of the upper() and lower() functions, however, does not alter the value of the string stored in the variable ‘animal’. You can prove this by printing the variable:

```
>>> print(animal)
```

elephant

The str() function

In some situations, you may have to print non-string characters such as integers or floats as a string character. Python provides the str() function to allow for conversion of non-string character to string character.

For example, assuming that the variable my_num holds the integer 85, you can use the variable as a parameter when you need to print the value stored in it.

Hence, to print the string ‘My employee number is 150’, you can enter the following statement:

```
>>> print("My employee number is " + str(my_num)+ ".")
```

The output would be:

```
My employee number is 85.
```

Again, converting the integer to a string using the str() function does not affect the original value stored in the variable my_num. You can prove this by applying the type() function on the variable.

```
>>> type(my_num)
<class 'int'>
```

Lists

A list is a mutable sequence type data. It is one of the most commonly used data type in Python. A flexible container, it can hold any type of data or items which may or may not be of similar kind. It can also store any number of data.

To create a list, you can assign values enclosed in square brackets to a variable.

To create an empty list:

```
item_list = []
```

To create a list, you will use the syntax:

```
my_list = [item1, item2, item3, item4]
```

To illustrate how list is used in Python, you can build the following lists:

```
list_1 = ["apple", "pear", "orange", "banana", 5, 8, 10, 3, 12.5, 6.25, 5.99, 6.75]
list_2 = [2, 4, 6, 8, 10, 12, 14, 16, 18]
list_3 = ["a", "e", "i", "o", "u"]
list_4 = [92.5, 25.8, 9.5, 2.10]
```

Accessing Items on a List

Lists are indexed like strings and they both start at zero. For example, if you want to access the first item on list_1:

```
>>> print(list_1[0])
apple
```

To access the third item on list_:

```
>>>print(list_1[2])
```

orange

To print the last item on the list, you can use negative indexing:

```
>>> print(list_1[-1])
```

6.75

Using len() function with List

The len() function is used to get the number of items on a list:

```
>>> len(list_1)
```

12

Removing Items from a List

The function remove() is used to remove an specified element from a list.

syntax: my_list.remove(value)

To illustrate, create the list my_list with the following elements:

```
>>> my_list = ['cat', 'dog', 'horse', 'monkey', 1, 8, 10, 15, 24.5, 1.75]
```

Assuming you want to remove the values ‘horse’ and the integers 1 and 10, you can use the remove() method thus:

```
>>> my_list.remove('horse')
>>> my_list.remove(1)
>>> my_list.remove(10)
```

To see the values left in my_list:

```
>>> print(my_list)
```

['cat', 'dog', 'monkey', 8, 15, 24.5, 1.75]

Take note that the use of the remove() function affected the elements stored in my_list. This is because a list is mutable.

Adding Items to a List

To add an item to a list, you will use the append() method.

Referring to the above example, you can add the string ‘python’ to my_list with:

```
>>> my_list.append('python')
```

To view the updated my_list:

>>> my_list

['cat', 'dog', 'monkey', 8, 15, 24.5, 1.75, 'python']

The string ‘python’ was added at the end of the list. Assuming you want to add a new string ‘elephant’ between the strings ‘dog’ and ‘monkey’, you will have to use the insert() method.

The syntax is: list.insert(index, object)

To insert ‘python’ between ‘dog’ and ‘monkey’:

>>> my_list.insert(2, "elephant")

>>> my_list

['cat', 'dog', 'elephant', 'monkey', 8, 15, 24.5, 1.75, 'python']

Sorting Items on a List

The sort() method is used to arrange or sort items of similar data type within the list. The syntax is: list.sort().

For example, create a list called num_list containing numeric values:

>>> num_list = [2, 10, 5, 3, 19, 4.5, 7.25, -2, 0]

To sort the elements in num_list in ascending order, you can enter the expression:

>>> num_list.sort()

Now, view the contents of num_list:

>>> num_list

[-2, 0, 2, 3, 4.5, 5, 7.25, 10, 19]

The elements of num_list are now arranged in ascending order. You can rearrange the items in reverse order by using the reverse method. Its syntax is:

list.reverse()

Hence, to arrange num_list in descending order:

>>> num_list.reverse()

>>> num_list

[19, 10, 7.25, 5, 4.5, 3, 2, 0, -2]

Tuples

A tuple, like a list, is a sequence type that contains an ordered collection of items. A tuple, unlike a list, is immutable.

The elements of a tuple are enclosed in parentheses. The parentheses are, however, optional and you can create a tuple without them. Tuples can contain items of different types. Like lists, a tuple can also store any number of items.

Creating a Tuple

Creating a tuple is as simple as assigning a group of items to a variable.

Examples:

mytuple_1 = (1, 3, 5, 7, 9, 11, 13) numeric

mytuple_3 = ("ruby", "gold", "diamond") strings

mytuple_2 = ("Python", 2.50, 7) mixed-type

Python likewise allows the creation of a nested tuple:

mytuple_4 = ("collection", (10, 20, 30), [2, 4, 6])

A tuple containing a single item may be created provided that a trailing comma is place after the item to indicate that it is a tuple and not a string.

mytuple_5 = ("members",)

It is also possible to create an empty tuple:

empty_tuple = ()

Here is an example of a tuple creation without the parentheses.

my_tuple = 7, 14, 21

Accessing Tuple Elements

There are several ways to access the items stored in a tuple.

- Indexing

The index operator [] is used to specify the corresponding index of the item that you want to access. Like other ordered data types, indexing starts from zero.

For example, create tuple_new which has strings as objects:

>>> tuple_new = ('m', 'u, 's', 'i', 'c', 'a', 'l')

To access the first item on tuple_new:

>>> tuple_new[0]

'm'

To access the fourth item:

>>> my_tuple[3]

'i'

To access the last item:

>>> tuple_new[-1]

'l'

- Slicing

The slicing operator is used to access a range of items in a tuple.

For example, create a tuple named my_tuple:

>>> my_tuple = ("p", "r", "o", "g", "r", "a", "m")

To access the elements from indices 2 to 4:

>>> my_tuple[2:5]

('o', 'g', 'r')

To access items from the beginning of the tuple to the fourth element:

>>> my_tuple[:4]

('p', 'r', 'o', 'g')

To access the fourth item up to the end of the tuple:

>>> my_tuple[3:]

('g', 'r', 'a', 'm')

- Changing the Elements of a Tuple

As stated previously, tuples are immutable so you cannot change its elements. The only time that you can change an item in a tuple is when the item is an element of a nested list.

For example, create a new tuple called my_tuple:

>>> my_tuple = (5, 1, 3, [5, 10])

To replace the item 5 (on index 3) with 15:

>>> my_tuple [3][0] = 15

To view the updated elements of my_tuple:

>>> my_tuple

(5, 1, 3, [15, 10])

- Deleting a Tuple

To delete a tuple including all elements stored in it, you can use the syntax:

del tuple_name

For example, you can delete my_tuple with:

>>>del my_tuple

- Dictionary

A dictionary is a container for unordered key-value pairs. The keys and values are separated by a colon and enclosed in curly braces {}. The keys are immutable and can only be a number, tuple, or string. The values are mutable and can hold any data type. A dictionary value may only be accessed by accessing its keys.

A dictionary's key-value structure is similar to those used in phone books, directories, menus, and other registries.

- Creating a dictionary

To create an empty dictionary, the syntax is:

dict = {}

To create a dictionary with key-value pairs:

dict = {key_1:value1, key_2:value2, key_3:value3, key_4:value4}

- Accessing the Elements of a Dictionary

To access a value in a dictionary, you will have to use its key. There are two ways of doing this: provide a specific key or apply the get() method.

To illustrate, here is a dictionary:

student_1 = {'Name':'Meryll Brooks', 'Age':16, 'Average Grade.':90.5}

To access the values contained in student_1:

>>> student_1['Name']

'Meryll Brooks'

>>> student_1['Age']

16

>>> student_1['Average Grade']

90.5

To access the values with the get() method:

>>> student_1.get('Name')

'Meryll Brooks'

>>> student_1.get('Age')

16

>>> student_1.get('Average Grade.')

90.5

- Adding Dictionary Entries

The assignment operator (=) is used to add a new key-value pair to a dictionary. Every time a key-value pair is added by assignment, Python evaluates if there is

a similar key in the existing dictionary pairs. If none, then it appends the new key-value pair. If a key already exists, it simply updates the existing value with the new value given.

syntax:

dict_name[key] = b

For example, if you want to add a new key-value pair to the student_1 dictionary above, you can enter the statement:

student_1['sex'] = 'female'

To display the updated dictionary, type the dictionary name on the >>> prompt:

>>> student_1

{'Name': 'Meryll Brooks', 'Age': 16, 'sex': 'female', 'Average Grade.': 90.5}

- Modifying Dictionary Values

To modify the current value, use the assignment operator to specify the new value to be associated with the key.

For example, to replace the value of the Average Grade:

>>> student_1['Average Grade'] = 85.0

Now, type student_1 to display the updated dictionary:

>>> student_1

{'Name': 'Meryll Brooks', 'Age': 16, 'sex': 'female', 'Average Grade.': 90.5, 'Average Grade': 85.0}

- Removing Entries from a Dictionary

You can remove a specified key-value pair from a dictionary and return the value of the deleted key.

For example:

To remove the key entry ‘sex’ from the student_1 dictionary:

>>> student_1.pop('sex')

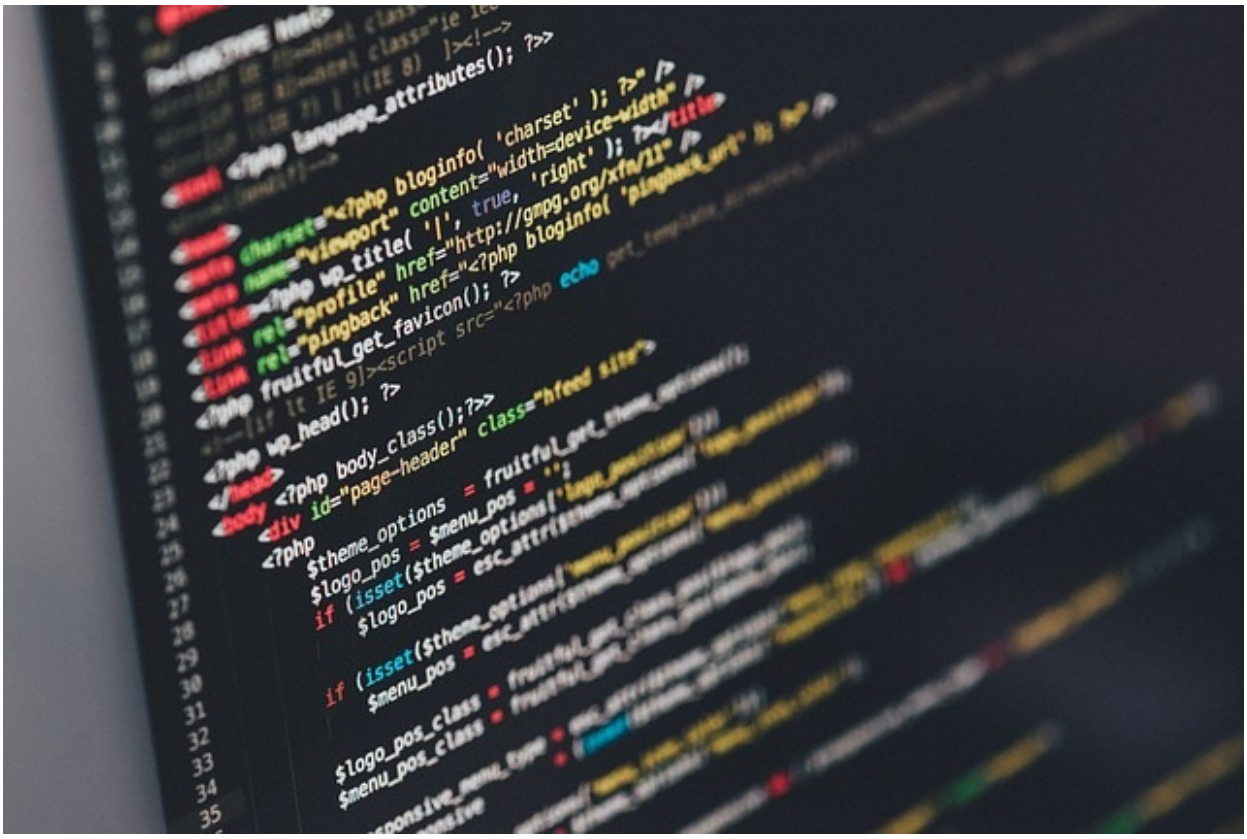
'female'

Now, view the updated dictionary:

>>> student_1

{'Name': 'Meryll Brooks', 'Age': 16, 'Average Grade.': 90.5, 'Average Grade': 85.0}

Chapter 4. methods and functions of python



A function is used to perform a particular action. It is a structuring tool that allows a code to be reused in different segments of a program. It promotes efficiency by minimizing code repetition and by breaking down complex programs into manageable segments. Functions are sometimes called methods, procedures, or routines. There are two categories of functions in Python: Built-in and user-defined.

Built-in Functions

Python offers several built-in functions. Here are some of the common ones:

abs() returns the absolute value of a number

dict() creates a new dictionary

float() converts string or number to a float

id() returns the object’s identity

type() returns the type of an object

input() reads a user’s input, converts it to a string, and returns it

print() print values to screen

int() converts another numeric type or string to an integer

len() returns the length or size of an object

list() returns a list

tuple() returns a tuple

max() returns the largest item

min() returns the smallest item

range() returns an iterable sequence

round() returns the rounded value of a float

str() returns a string version of an object

User-Defined Functions

statement(s)

A function's body must have at least one valid statement. Multiple statements must use similar indentation in order to form a block.

return statement

A return statement returns a value from the function. A function with no return statement returns None.

Modules

Here is a simple function that greets the person named and has no return statement:

```
def greeter(name):
"""This function greets the name supplied as parameter."""
print ("Hello, " + name + ", Have a nice day!")
print(greeter ('Marthy'))
print(greeter ('Shane'))
print(greeter ('Jamille'))
```

Here's the output:

Hello, Marthy, Have a nice day!

None

Hello, Shane, Have a nice day!

None

Hello, Jamille, Have a nice day!

None

Following is a function with a return statement:

```
def multiplier(x, y):
"""Function to get the product of two numbers."""
return x * y
```

To use the multiplier function, you can call it on the >>>prompt by using the print() function and providing two numbers as parameters such as:

```
>>> print(multiplier(5,7))
35
>>>print(multipler(9,7))
63
```

Modules

It contains statements, definitions, and codes. They are used to organize large programs into smaller and more manageable files. Modules promote code reusability which makes programming more efficient. You can save a lot of time and space by putting together your function definitions in one module and just importing it in your program instead of copying their definitions each time you need them.

To illustrate the concept, create a module containing definitions of functions called ‘adder’ and ‘multiplier’ and save it as calculator.py:

```
def adder(x, y):
"""This function adds the given values and returns the result """
total = x + y
return total
```

```
def multiplier(x,y):
"""This function multiplies the given values and returns the result """
product = x * y
return product
```

Importing a Module

To import a module, you will use the keyword ‘import’. To access the definitions in a module, you can use a dot operator between the module name and the function you want to access. For instance, to import the calculator.py module and use the stored definitions:

```
>>> import calculator
>>> calculator.adder(5, 9)
14
>>> calculator.multiplier(2,4)
8
```

Python’s Built-in Modules

Python has several built-in modules that you can import for use in your programs. Some of the most commonly used ones are the math module, random module, time module, and calendar modules.

Math Module

Importing Python’s math module allows users to access its attributes, constants, and mathematical functions such as pi, square root, sine, cosine, and gcd. After importing the math module, you can use the dot operation on math and specify a method or attribute after the dot.

For example:

```
>>>import math
>>>math.sqrt(64)
8.0
>>> math.gcd(14, 7)
7
>>> math.pi
```

3.141592653589793

Displaying the Contents of a Module

You can view the methods and attributes after importing it by using the built-in function dir() and providing the module's name as the argument.

For instance, to display the contents of the math module:

```
>>>import math

>>>dir(math)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp',
'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
```

Managing External Files

So far, you have used the keyword import to access external files for use in your current program. You will notice that import statements were simple statements that do not specify the directory from which the external files are to be imported.

Consider this import statement in a previous example:

```
import calculator
```

When working with Python, there is always a current working directory in memory. When you enter an import statement, the interpreter will look for the file in this directory.

Python handles files and directories with its os module. If the file you want to access is stored in a folder other than the current working directory, you will have to change the current directory with the chdir() method.

For example, assuming that you want to import a file named collection.txt which is stored in a folder named 'Python/Personal' which is not the current directory, you will need to use these statements:

```
#Import the OS module

import os

# Change the directory to "/Python/Personal"

os.chdir("/Python/Personal")
```

import collection.txt

Using the getcwd() Method
syntax:

os.getcwd()

Example:
import os

This displays the location of the current working directory

os.getcwd()

Conditional programming

The ability to work with data and variables would be useless if you cannot deal with that data on a conditional basis. Fortunately, like all programming languages, Python has numerous methods of doing just that.

Some of the options for conditional statements available in Python are,

If...then conditionals

The basic idea is that if something is true then we execute a specific set of instructions, otherwise we do not.

The simplest form of this is as follows:

if (x==1): do something

In this example, we simply state that if x equals 1 then we do something.

'Do something' as shown here would represent a short action that can fit on a single line. In this format, the next line will indent to the same column as the **if** statement, which indicates that code will be executed regardless of the result of the **if** statement.

In a more complex situation, after the colon, a return is entered and the next line will indent.

All subsequent non-empty lines will be part of the **if** conditional, and will be indented to the same level.

Entering an empty line will drop the indent out of the conditional statement.

```
if (x==1):
        Do thing 1
        Do thing 2
        Do thing 3

    Next Code to Run
```

The conditional within parenthesis shown is testing a simple equality. The conditional can be more complicated than this.

Essentially, whatever conditions are defined within the parenthesis will be evaluated as true or false. If it evaluates to true as a whole, then the conditional statements will be executed, otherwise they will not.

Following is a slightly more complicated example:

```
>>> x,y=0,10
>>> if (x==0 and y==10):
      print(True)

True
```

The basic **if...then** conditional only allows us to execute specific code if the condition is true.

That does not mean we cannot execute code in the negative, only that the negative has to be true.

In other words, if we want to execute code if x does not equal 0 then we have set up the conditional so that when it is true that x does not equal 0 then the code is executed like this:

```
if (x<>0): Do something
```

In order to run code specifically when the conditional is false (when x==0 in the case above) we need to go to the **if...then...else** statement.

```
if (x<>0):
        do something # when true
    else:
        do something else # when false
```

The indenting/formatting rules following the else are the same as the initial if statements so all non-empty subsequent lines are block indented and considered part of the else block. These will all be executed if the conditional is false.

If … elif... else adds one more level of sophistication. In this case, elif stands

for else if.

The else if combines the else and an additional if so it is only executed if the preceding if statement is false and the conditional it is set to test is True.

```
if (x<0):
        do something # when x<0
    elif (x==0):
        do something else # when x is 0
    elif (x==1):
        do something else # when x is 1
    else:
        do something when all the above are false
```

It is possible to nest these statements as well.

When nesting if statements it can get difficult to follow your own code so care must be taken to maintain proper logic.

```
if (x==1):
        do something
        if (y==1):
            do something
    else:
         do something else if x<>1 - note that the result of (y==1) has
nothing to do with this because the else is indented to the if (x==1)
level.
```

Many languages include another conditional type.

The **switch/case** statement which allows you to test a conditional for multiple specific values rather than just true or false.

Unfortunately, Python does not have such a statement. Similar functionality can be accomplished using dictionaries or extensive elif statements but neither will offer identical function or equal ease of use.

Loops

It is the ability to repeat or loop the program through a specific set of tasks.

All programs, other than those that perform a specific task and exit, contain at least one loop. Typically, this is the main loop, where the program continually loops waiting for user or object input for it to act upon.

Additionally, loops can be used to apply repetitive processes on objects using a compact set of instructions.

Python offers two types of loops.

The **for** loop and the **while** loop.

They both allow for the repetitive looping through a specific operation, but differ in how they test if the loop should continue to be processed.

For Loop

In the **for** loop, the loop will execute over a given range of items.

For example:

```
>>> for x in range(0,3):
    print(x)
0
1
2
```

First off, the syntax is similar to the **if** statements we covered earlier.

The **for** loop is ended with a colon.

With a single instruction in the **for** loop, we can leave that instruction on the same line immediately after the colon. A return after that will indent to the next instruction outside the **for** loop.

If the for command contains multiple instructions, start those instructions on the next line. They will be indented to show they are contained within the loop.

A blank line entered at the end will close out the loop and drop the indent down from the loop for the next instructions outside the loop.

Range Command

Next, we have the **range()** command.

The range command is a built in python method used to generate a range of numbers and is not explicitly part of the for loop syntax.

It is just there to make the numbers to iterate through. Like other ranges in python, the numbers provided do not represent a range from 0 – 3. They represent outputting three numbers starting at 0, which gives us 0, 1, and 2.

3 is not output in the range.

This is slightly different than the for loops you might be accustomed to in other languages. In those, a given range generally included the last number defined – for x=1 to 10 for example would output 1 – 10, not one through 9.

While we use the range() function to generate the numbers to iterate over in the for loop, it is by no means the only option.

We can actually pull numerical values from any iterable source to iterate over that source.

For example, we can use the length of lists, tuples or strings to process through that item.

```
>>> x=['apples', 'oranges', 'bananas', 'peaches', 'plums']
>>> for y in x:
    print(y)

apples
oranges
bananas
peaches
plums
```

It is important to note in this case that we are actually iterating sequentially through the list values and setting y equal to those.

We are not setting y to the position values of 0 through 4.

If we had used print(x[y]) - or print the string at the position y in the list - we would get an exception stating that y cannot be a str.

To iterate through a list using the position number we would have to set y to a numeric range and iterate through it that way.

```
>>> for y in range(0, len(x)):
    print(x[y])

apples
oranges
bananas
peaches
plums
```

Going this route makes it easier to get every x value within the list.

The range method includes a step value attribute that allows us to step through the list in spaces other then 1.

For example, to get every other item in our list we would use:

```
>>> for y in range(0, len(x), 2):
    print(x[y])

apples
bananas
plums
```

Or, get the list backwards:

```
>>> for y in range(len(x)-1, -1, -1):
    print(x[y])

plums
peaches
bananas
oranges
apples
```

While Loop

The **while** loop continues while a specified conditional is true.

For example:

```
>>> y=1
>>> while y<=10:
    print(y)
    y+=1

1
2
3
4
5
6
7
8
9
10
```

In this case, the loop will continue as long as y is less than or equal to 10.

Unlike the **for** loop, which is specified to execute over a prespecified range and then exit, the while loop will continue as long as the specified condition is true.

Therefore, while loops are susceptible to becoming infinite loops – or code that loops forever.

To prevent that, it is imperative that code be included within the loop that will make the condition false at some point.

In this case, y+=1 increments y by one each time the loop executes. When y becomes 11, the conditional becomes false and the loop ends.

Break Command

There is one additional way of gracefully exiting a **for** or **while** loop.

The **break** command forces the loop to terminate prematurely.

When coupled with an **if** statement, **break** can be used to conditionally and gracefully exit a loop early.

In most cases, choosing which loop structure to use and properly designing the code within the loop will negate the need for the break statement so it is generally considered better if you can avoid it.

There are some cases however where it can be necessary.

Chapter 5. :processing data analysis and algorithms

When it comes to data analysis, it is not as easy as it looks. One of the reasons why it is difficult is because only a few people have mastered the art of data analysis. This means that only a few people can explain how it is done.

Surprisingly, many people try to analyze data daily, but the majority fails in their efforts. This is because experts in this field haven't taken time to explain how they think while analyzing data.

Data Science is an art. It is not a concept that one can teach a computer. Data analysts use different tools to achieve their task, right from linear regression to classification trees. Even though all these tools are known to the computer, it is the role of the data analyst to figure out a way in which he or she can gather all the tools and integrate them to data to develop the correct answer to a question.

However, the process of data analysis has not been written down properly. While there are many books written about Statistics, none of them tries to address how one can create a real-world data analysis solution. The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

On the other hand, coming up with an important framework involves classifying elements of data analysis using an abstract language. In some cases, this language might be mathematics. Conversely, the same details of the analysis are what make each analysis complex and interesting.

The Cycle of Analysis

You might look at data analysis and think that it follows a linear, step-by-step process that has a well-developed result. However, data analysis is an iterative and non-linear approach that is depicted by a series of epicycles. In this approach, information is learned at each step which then decides whether to redo and refine the next step that is already performed or proceed to the next step.

When it comes to analyzing data, the iterative process is used in all steps of the data analysis. Besides this, certain data analysis might appear fixed and linear because of the algorithms encapsulated in the different software.

Therefore, it is important that one understand what it means by the term “data analysis”. Although a study of data involves creating and implementing a plan for gathering data, data analysis assumes that data is already gathered. Most importantly, a study will involve the creation of a hypothesis, designing of a data collection procedure, gathering of data, and interpretation of the data. However, since data analysis assumes that data should be collected already, it involves the development and refinement of a question and process of analyzing and interpreting data.

There are Five Major Activities of Data Analysis:

1. State and refine the question
2. Explore the data
3. Create formal statistical models
4. Interpret the results
5. Communicate the results

All the above activities happen on different occasions. For instance, it is possible to go through all of them in a day but handle each in detail in a period of more than one month. So, let’s look at the overall framework applied in each of these activities.

While there are many different types of activities that one can engage in while performing data analysis, each aspect of the whole process can be undertaken through an interactive process. Most importantly, for each of the above five activities, it is advised that you include the following steps:

1. Define or set the expectations

2. Collect information and compare the data to your expectations
3. If the data fails to match your expectations, revise or fix the data so that both your expectations and data match

Going through all the three steps above is what is referred to as the cycle of the data analysis. While you navigate through every stage of the analysis, you will be required to go through the epicycle to constantly revise your question, formal models, interpretation, and communication. A repeated cycle through each of these five major activities forms the largest part of data analysis.

Define the Expectations

In this step, you intentionally lay down what you expect before you can do anything such as performing a procedure, inspecting your data or typing a command. For the experienced data analyst, creating expectations might be automatic or a subconscious process. Despite this, it is important to think about it. For instance, if you are going to shop with friends and you have to stop by an ATM to withdraw some money, you need to decide on the amount of money that you want to withdraw. You need to have some expectations of the price of the things you are going to buy. This could be something that you have no problem with if you know the price/s of the product/s you are going to buy. This is an example of prior knowledge. Another example of prior knowledge would be to know the time that a specific restaurant closes. Using that information, you can schedule your time and activities so that you show up for dinner before it closes.

You can also find out additional information from your friends that will help you come up with expectations or Google a restaurant to learn more about their working hours. This procedure that you apply on prior information to develop expectation or implement an analysis procedure is the same used in every main activity of the analysis process.

Information Collection

This step requires one to collect information related to the question or data. For questions, one collects information by doing a literature research or finding out from experts. For the data, once you have developed some expectations about what the result can be when the data is inspected, it is okay to go ahead and carry out the operation. The results of this activity include data that you need to collect and determine whether the collected data matches your expectations.

Comparison of Expectations

Once you have the data in your hands, the next step is to compare your expectations to the data. Here, there are two possible results:

1. Your cost estimations match the amount on the check
2. Your cost estimations fail to match

If both the cost estimations and amount match, then you can move on to the next activity. Alternatively, if your expectations cost 60 dollars, but the check is 30 dollars, then your expectations and data are different. In this case, there could be two possible reasons for the difference: the first is that you may have wrong expectations and you need to revise; and second, the check may be wrong and contains errors. One key indicator that can hint on the status of your data analysis is the easiness or difficultness to match the data you collected to your original expectations.

Volume, Velocity, and Variety

Big Data has various “V”s. The major ones include velocity, variety, and volume. Big Data surpasses the storage capacity of normal databases. The scale of data generated is massive. As of today, a huge amount of data is generated. One reason for this is because of the increase in interaction. Interaction is a new phenomenon besides just transaction of data. Data interaction comes from activities of the browser, personal digital recorders, and geo-location.

With the advent of the “internet of things”, massive data is produced that humans spend their entire time trying to analyze.

A good Data Scientist should know how to control volume. He or she should know how to create algorithms that can intelligently use the size of the data effectively. Things acquire a new direction when you have gargantuan data because each similarity becomes important, and one can easily make false conclusions. In most business applications, extraction of correlation is enough. However, the right Data Science uses techniques that determine the cause based on these correlations.

Data velocity will always accelerate. There is an increase in Facebook posts, tweets, and financial information generated by many users at a higher speed. Velocity increases the volume of data and reduces the time of data retention. For example, a high-frequency trading activity depends on data streams and fast information. But the authenticity of the data reduces rapidly.

Lastly, data variety has gone deep. Models which depend on just a handful of variables can now produce hundreds of variables because of the increase in computing power. The rate of change in volume, velocity, and data variety is currently possible for new economic-metrics and various tools.

Machine Learning

Machine Learning refers to how systems learn from various types of data they process. It is possible to train a system based on particular data to make decisions. The training process occurs continuously to enable systems to make updates and enhance decision-making ability. Systems that use spam filters are a great example to demonstrate how Machine Learning is applied. These systems

use a Bayesian filter to change decisions.

Therefore, it will continue to stay ahead of spammers. The ability to dynamically learn is important because it helps prevent spammers from gaming the filter. Credit approvals use neural-nets and are a great example of Machine Learning technique. Besides that, Machine Learning prefers data compared to judgments. Hence, a good Data Scientist should have a variety of both. Machine Learning has helped in finding answers to questions of interest, and it has further proved to be a game-changer. What makes Machine Learning very interesting is the four characteristics of machine intelligence:

1. It is built on a strong foundation of a theoretical breakthrough
2. It redefines the current economic paradigm
3. The final result is commoditization
4. It unearths new data from Data Science

Supervised and Unsupervised Learning

There are two broad ways that a system can learn: supervised and unsupervised learning.

Supervised learning is where a system makes decisions depending on the type of data entered. Automated credit card approvals and spam filters apply supervised learning to achieve their functions. The system is supplied with a historical data sample of outputs and inputs. Based on this type of data, the system establishes the relationship between the two using Machine Learning techniques. You will need to use your judgment to choose the best technique to handle the task.

Unsupervised learning happens when you only have input data (X) without a corresponding output variable. Unsupervised learning aims to build a model of the underlying structure in the data order so that you can learn more about the data. It is called unsupervised learning because there is no correct answer and teacher. Algorithms are left to decide and discover interesting structure in the data.

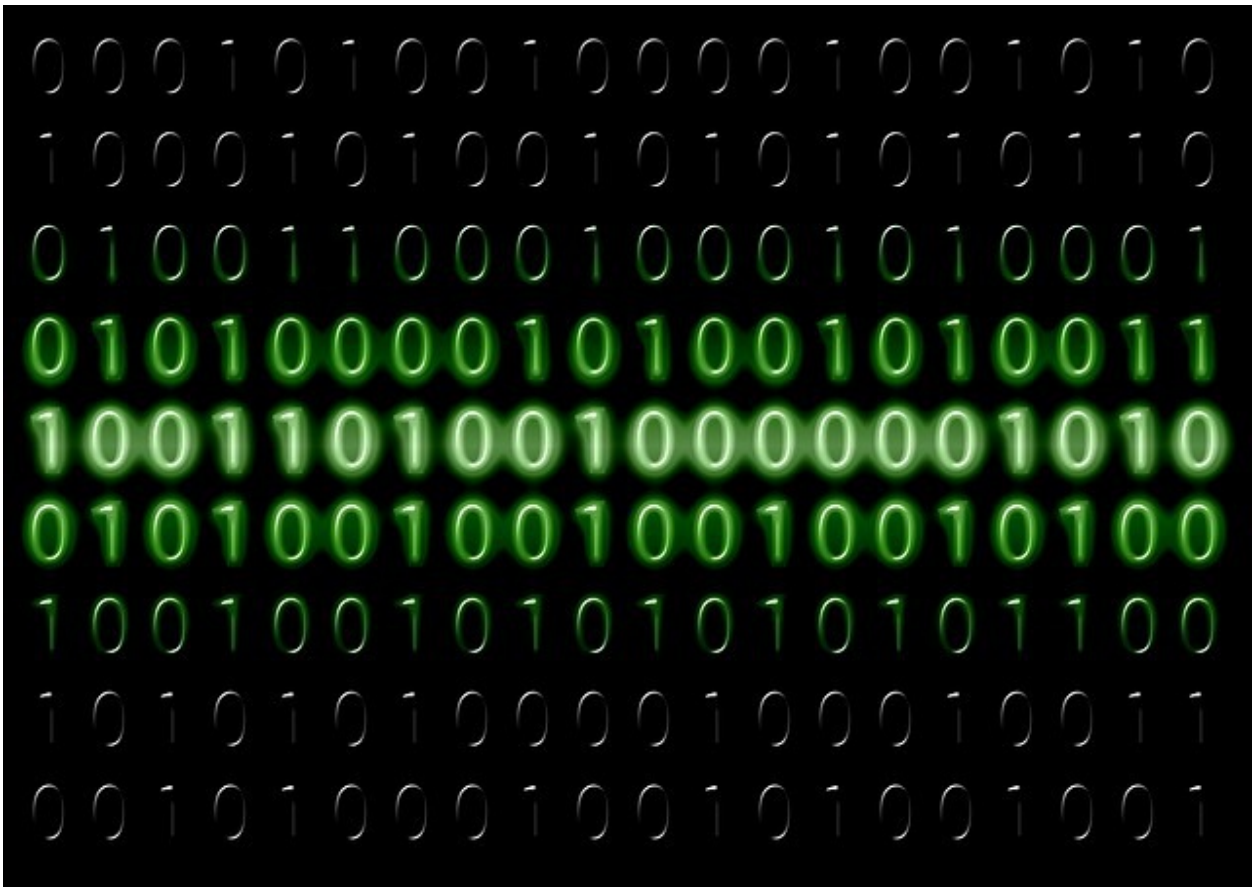
Cluster analysis is an example of unsupervised learning. Cluster analysis selects a group of entities each with a different attribute and divides the entity space based on how far or near the entities of the attributes are. This will rearrange and redefine data by labeling it using additional tags. Factor analysis is part of the unsupervised learning technique.

Predictions and Forecasts

Data Science involves making forecasts and predictions. However, there is a distinction between the two. Predictions focus on highlighting a single outcome. If a person says that “it will be cold tomorrow,” he or she has predicted. But if they say that “the chance of tomorrow being cold is 40%,” they shall have made a forecast. This is because a forecast provides outcomes in the form of

probabilities.

Chapter 6. Object oriented programming in python



Object-oriented programming (OOP) is a model in which programs are organized around objects or data without the use of functions and logic. An object has its unique behavior and attributes. In object-oriented programming, the historical approach to programming is opposed while the stress is given to how the logic is written rather than defining the data within the logic. Examples of objects range from physical entities such as humans to small programs like Widgets.

A programmer focuses on the first step known as data modeling in which all the objects are identified to be manipulated, and these objects relate to each other. The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

In OOP, the developers focus on object manipulation, rather the logic required to manipulate them. This approach is well-suited to programming for the programs,

especially the complex, larger, and actively maintained programs. The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

Open-source organizations also support object-oriented programming by allowing programmers to contribute to such projects in groups that results in collaborative development. Furthermore, the additional benefits of object-oriented programming include the code scalability, reusability, and efficiency.

Principles of Object-oriented programming: There are many principles involved in the object-oriented programming. Here are some of them discussed below:

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

Encapsulation

The state and implementation of each object are held inside a defined class or boundary but privately. Other objects can only access to this class by calling a list of public functions or lists. Else, the objects cannot access to this class or the authority to make changes. Such programming characteristics of data hiding avoid unintended data corruption and provide greater program security.

Inheritance

The object-oriented programming (OOP) ensures a higher level of accuracy and reduces time development. Another property of object-oriented programming results in more thorough data analysis. A relation and subclasses build between the objects which can be assigned and allow developers to reuse a common logic while maintaining the unique hierarchy. This property results in more thorough data analysis, high accuracy, and save time.

Abstraction

The objects reveal the internal mechanisms only. This can be helpful and relevant for the use of other objects. Due to this, the concept of a developer builds that is supportive of going for more addition or making changes over time more easily.

Polymorphism

Depending on the context, objects are allowed to take on more than one form. It is the program that will determine the meaning and usage for each execution of an object, cutting down on the need to duplication code.

Criticism on Object-oriented programming

Developers criticized the object-oriented programming due to multiple reasons. One of the major concerns about object-oriented programming is that it does not focus on computation or algorithms. Object-oriented programming codes may be more complicated to write and take longer to compile. But developers also find alternatives for such complications. The alternatives include the following:

- Functional programming
- Imperative programming
- Structured programming

However, only the most advanced programming languages enable developers with the options to combine them.

Python as an Object-oriented programming language

Python programming language is widely used as an object-oriented programming language for web application development. According to a survey, 90 percent of the programmers prefer to work with Python language over other languages due to a lot of reasons. Its simplicity, readability and easy interfacing are the major reasons for its preference. Python is used in object-oriented programming as well as follows a procedural paradigm; hence the advanced and diverse applications come out with clean and super simple codes.

Application development with Python programming language also requires some frameworks with the help of which application development is easier for developers. The most used frameworks include Django, CherryPy, Pyramid, Flask, Qt, PyGUI, and Kivy, etc. The use of these frameworks is based on the nature and the requirements of individual projects. The assistance is provided by these Python frameworks to build sophisticated applications with minimal efforts and time.

Python is a popular scripting language for many software development processes. Furthermore, Python can be economically utilized to integrate disparate systems together.

Classes and Objects

It is about the basics that come with the Python code, it is time to move on to our first topic of working with classes and objects. These two topics are important because they will make sure that everything in your code stays in the right place. Your objects are the components which will help define certain parts of the code so that they are organized and easier to understand. The objects all share something in common, and they will be placed into the same class, helping your code to work the way that it should.

There are a lot of things to understand when it comes to working with objects, but the most important part is that whenever you place some objects into the same class, they should have something in common. You can place anything in the same class, but the best way to do this is to place ones that are similar together in the same class. It adds in a little more order to the whole thing. Think about this process as you organizing the closet. You have to make sure that your closet is organized by placing the shoes in one area, hanging up the clothes in another, adding the purses together, and then having a place for all those extra things lying around. You do this process task it ensures that everything looks nice and is easier to find later on.

This is the same idea when you are working on classes. You want to place the similar objects into the same class so that it is easier to find them later on. The program will work more efficiently when these objects are similar, rather than randomly placing unrelated objects into the same class.

To make things easier to understand, you should see objects as different aspects and parts of the code that you are writing and then the classes will be like the containers which you will use to store away these objects. You can easily create a class out of anything that you want and then add in similar objects that match up together into that class. You can pick out whatever label that you want to go with the class, just be certain that it actually makes sense for the type of object that you want to place inside the class. When someone looks at the class, they should have a good idea of what you placed inside of it and the objects should all match that label as well.

If this is a new idea to you and you are just getting started with learning a new coding language, objects and classes are a good place to start because they will help you to learn more while also making sure that everything will stay organized in the code that you are working on. It is your job to learn how to create these classes and then get the objects to go inside of them so that the code

works the proper way. Let's take a look at how that is going to work in Python.

How to create a new class

Now that we had a little introduction to these objects and classes, it is time for you to learn the steps on how to create your own class. You have to do this because you will find that it is hard to get started with anything in Python if don't do this first. Once you are ready to create the statements that are needed for these classes, you should create a new definition.

You need to place the right keyword first to get this going and then add the name of the class (the name that you are giving to it) right afterward. This will then be followed with a 'superclass' that you will place inside of parenthesis. Another thing that you should consider is that at the end of your first line in Python, you need to add a semicolon. Your code will work without it, but this is considered a part of coding etiquette, so you have to make sure that it is present.

myGenericVehicle.Display_Vehicle()

Before we move on, take the time to open up your text editor and write this code down so you won't forget it. As you work on writing this out, you should notice that there are a few of the basics that we discussed and explained earlier that will show up in this code. You will also see the definition of the object and the method, the destructor function, the different attributes that you will need, and also some regular functions. Since this is the first code that we will have a look at, let's break it down a bit so that you are able to understand and know how to write out codes in this language better.

Class definition

This is where you need to write out the object instantiation and then the class definition so that your syntax is proper when put inside the code. These are important because they are the part of the code that will tell your compiler what it should do and which commands are important. If you want to invoke the new class definition inside of the code, you just have to use the 'object.attribute' or the 'object.method()' function and it will work flawlessly.

Special attributes

The next thing that you can focus on in the above code is the special attributes that you want the code to recognize for you. These special attributes will give you some peace of mind because you can know when the attribute will be seen and you can make sure that it is used in the right way to prevent the code from

getting messed up. There are a lot of good attributes that can be used in Python, but some of the ones that you really should pay attention to as a beginner include:

__bases__

This is considered a tuple that contains any of the superclasses

__module__

This is where you are going to find the name of the module, and it will also hold your classes.

__name__

This will hold on to the class name.

__doc__

This is where you are going to find the reference string inside the document for your class.

__dict__

This is going to be the variable for the dict. Inside the class name.

How to access members of a class

Now we need to look at how you are able to access members of a class. To make sure that your text editor and compiler are able to recognize a class and execute the parts that you want, you must make sure that this code is properly set up to access all the members of that class. You will be happy to know that there are a few options you can use to make this happen, and all of them will work out well, but the 'accessor' method is the most popular and the most efficient one to work with.

Make sure to open up and place this in your compiler before we move on. If you ask the compiler to run this, you should get some results that show up right away. The results that you get will include that the name of the cat is Frisky (or whatever name you would like to add in there) and that the age is five while the weight is 10. This is the information that we placed into the code, but you can add different names and different numbers in there if that works out better for you. Try out a few different options to see what will work and learn how you are able to make it work for you.

As you can see, classes and objects are really easy to work with. You can create

the class and then add the objects that you want and then match it into the class that you created. This will keep your coding nice and organized just like the way that you want it, and it also makes sure the program will actually work.

Chapter 7. Teaching you the python language

When working with joint processing in the Python programming language (in particular, CPython, the reference implementation of Python written in C) is GIL. The GIL mechanism is a deadlock mechanism that protects access to Python objects, preventing the simultaneous execution of Python byte codes in multiple threads. CPython uses a reference counter to implement its memory management. This results in the fact that multiple threads can access and execute Python code at the same time; this situation is undesirable and it can cause incorrect data processing and we say that this type of memory management is not oriented towards multi-threaded management. To solve this problem, there is a GIL, as its name indicates, a certain lock that allows only one thread to access the code and Python objects. However, this also means that in order to implement programs with multiple threads in CPython, developers must be aware of the GIL and circumvent it. It is for this reason that many who have problems implementing co-processing systems in Python.

Why python

The huge number of development tools available in Python has prompted an

increasing number of developers to start programming in Python, which makes Python even more popular and easy to use; I call it the vicious circle of Python . David Robinson, DataCamp's research supervisor, wrote a blog post about Python's incredible growth and called it the most popular programming language.

However, Python is slow, at least slower than other popular programming languages. This is because Python is a dynamic typing language that interprets in which values are not stored in tight buffers, but in scattered objects. This is a direct result of the readability and usability of Python. Fortunately, there are various options for making your Python program run faster, with concurrency being one of the most difficult among them; and this is exactly what we are going to master in this book.

Customizing Your Python Environment

Before we move on, let's go through a number of descriptions about how to configure the necessary tools that you will use in this book. The process of obtaining a Python distribution for your system and a suitable development environment, as well as how to download the code that is used in the examples contained in all chapters of this book.

General installation

Let's look at the actual process of obtaining a certain Python distribution for your system and the corresponding development environment:

All developers can get their own Pyton distribution from https: / / www.python. Org / downloads / .

Even though both versions of Python 2 and Python 3 are supported and maintained, throughout this book we will use Python 3.

For this book, a flexible option would be an IDE (integrated development environment). It is possible to develop Python applications with a minimal text editor such as Notepad or TextEdit, it is usually much easier to read and write code using an IDE purposefully developed for Python. They include: IDLE , PyCharm , Sublime Text, and Atom .

Now you have learned the basics of the concepts of joint and parallel programming. All this relates to the design and structural programming of

commands and instructions so that different sections of your program can be executed in some kind of effective way, and at the same time share the same resources. Since when some commands and instructions are executed at the same time, time is saved, parallel programming provides a significant improvement in the execution time of the program when compared with traditional sequential programming.

However, when designing a parallel program, various factors should be taken into account. While there are special tasks that can easily be divided into independent sections that can be executed in parallel (stunning parallel tasks), others require different types of coordination between the existing program commands so that such shared resources can be applied correctly and efficiently. There are also tasks with sequential processing that is intrinsically intrinsic to them, in which no joint processing and parallelism can be applied to obtain program acceleration. You should be aware of such fundamental differences between these tasks so that you can carefully design your co-processing programs.

In recent times, a paradigm shift has occurred that facilitated the implementation of joint processing in most aspects of the existing programming universe. Now joint processing can be found everywhere: desktop and mobile applications, video games, web and Internet development, AI and so on. Collaborative processing is still growing and is expected to continue to grow in the future. Therefore, it is extremely important for any experienced programmer to understand the joint processing and related concepts, as well as to know how to integrate these concepts into their applications.

Python, on the other hand, is one of the most (if not the most) popular programming languages. It provides options in most programming sub-sectors. Combining collaborative processing and Python in this way is one of the most important topics to learn and master in programming.

If we go through the market details, we have so many programming languages available and have been used in companies today. However, among these programming languages, Python is a high-level programing language. The structure of this language is well suited for many purposes and enables developers and programmers to express important concepts and necessary information in few and lesser words as compared to Java and C++. Python has the ability to support structured programming as well as to object-oriented programming. The language could also be incorporated into existing applications with requirements of programmable interface.

The Semantics of the Language

So the robust programing language Python is considered as greatly readable programing language. It is characterized and recognized by a visual layout which is uncultured and makes frequent and repeated use of English keywords. On the other hand, other programming languages make use of not English Keywords but use punctuation. Compared to the C language, it has a smaller number of syntactic exception.

Uses of Python in the Industry

Python is the popular programing language, especially in the web development field. Though predominantly and to a great extent used in the back end, it has great integration with database, good restatement time and some good number of web standards has been the reason of its fame in the industry. One of the famous question-answer platform Quora uses many python codes.

It is Python which is used to write applications with large size. A website like Google and NASA and YouTube make wide use of Python and specifically for its applications for smartphones. In addition, online transactions system in the US that is the New York Stock Exchange (NYSE). Some other ways and modes of using Python are:

- Making of RSS reader
- MySQL
- Calendar creation in HTML
- Working with files
- Foe online applications, programing CGI

In addition, it is yet a famous tool that is widely used by software developers for testing and checking build control as well as management. Moreover, for task integration, Python is the language to be used by companies. We can take the example of 3D software Maya that makes use of Python for both complex integrations and for automating small user tasks.

Applications of Python

Languages like Python might not always be used to engender a unique product, but of course, it is used to power the tools which are responsible for the creation of such kind of products. For example, look at the companies with a smart

internal system in use. They do not need compiled languages.

Moreover, the language has found diverse applications in companies that develop software and games, language development, web framework, and many more. This gives Python an edge over the rest of the programing languages, and that is the reason for its success in the industry. Let’s dive into some of the advantages Python has in the industry.

- Provision of support for large standard libraries that are equipped with a number of operations like web service tools, internet, protocols, and operating system interfaces.

- It is also used to integrate EAI Enterprise Application Integration that makes it possible and convenient to pave the way of development of Web services like COBRA or COM components by invoking them. Its control capabilities are very powerful and call directly through C++, C, and Javascript via Jython.

- It yet processes XML and other languages because it can easily run and operate on a modern system of operations using the same byte.

- It has the capabilities to improve programming productivity. The clear and object-oriented design has increased from two to ten-fold productivity of the programs while making use of languages like Perl, Java, C, and C++.

- The integration features are way too strong, and its unit testing and enhanced control abilities which results in increased speed for the productivity of most of the applications that use Python and the important thing is scalability which makes this language a great option for protocol network applications.

- In the current era, big data is becoming extensively important for big corporations, and especially when it comes to businesses. So, when it is about data processing, data scientists prefer to use Python as a tool for the operations. This is the sole reason that Python has become the best PL to be used when it comes to big data.

- So for building analytical tools, data scientists find Python the best fit. Moreover, since data is something that makes you money in the current era, the use of Python could not be ignored and should be incorporated in your execution plan.

- It is especially more desired for machine learning, and this is one of the leading businesses around the world right now. Advanced machine learning

could be further used for a huge number of tasks in the business.

- Finally, we should be talking about the cybersecurity which is very important in the current digital era of the world where we have these money exchanges gone digital, and this is very important to have the cybersecurity very strong to reflect hacking attempts. According to a survey about the cybersecurity performance of Python, it is a clear winner in the industry right now.

In addition to all the aforementioned stuff, it is yet another great feature of the programing language that Python is completely free and you do not need to pay a penny for it. Moreover, from this, it means that the expansion set of supporting modules and tolls and libraries are free. A wide number of its integrated development environments are free like Eclips, Pydev, and PTVS and Spyder Python and more.

So these are some of the reasons which have made Python the best fit for businesses and in the industry. I thin without Python; these big tech giants would not have been so much able in terms of influence and of course, monetary power. It is going to give rise to the nest IoT boom in days to come and most importantly in machine learning.

The Python coding language is one of the best options that you can choose to use when it comes to programming, especially when you want to work with machine learning. Python is a very simple coding language to learn, and it is often the first one that beginners will take a look at because of its simplicity. But don't be fooled. Just because it is simple to learn Python doesn't mean it won't have the power and strength that you are looking for in a programming language.

The first thing that you will want to do after installing the Python program from www.python.com is to download the proper IDE. This is a great environment to use with Python and will give you all the help that you need. It also includes the area to do a Python installation, your editors and your debugging tools. This IDE is easy to install, light, and it has a lot of great development tools that you will enjoy. It also has its own command line utility so you can install some third-party software as needed. And with this IDE, you won't have to go through and install the Python environment separate.

In order to download the Anaconda IDE, there are a few steps to complete. We are going to look at the steps that you can use when installing this for a Windows computer, but the steps for installing on a Linux or Mac computer are pretty much the same:

1. To start, go to https://www.anaconda.com/download/
2. From here, you will be sent to the homepage. You will need to select on Python 3.6 because it is the newest version. Click on "Download" to get the executable file. This takes a few minutes to download but will depend on how fast your internet is.
3. Once the executable file is downloaded, you can go over to its download folder and run the executable. When you run this file, you should see the installation wizard come up. Click on the "Next" button.
4. Then the License Agreement dialogue box is going to appear. Take a minute to read this before clicking the "I Agree" button.
5. From your "Select Installation Type" box, check the "Just Me" radio button and then "Next".
6. You will want to choose which installation directory you want to use before moving on. You should make sure that you have about 3 GB of free space in the installation directory.
7. Now you will be at the "Advanced Installation Options" dialogue box. You will want to select the "Register Anaconda as my default Python 3.6" and then click on Install.
8. And then your program will go through a few more steps and the IDE will be installed on your program.

As you can see, setting up an environment in Python can be simple and doesn't have to be very complicated to use. You can choose to work with other forms of IDE as well and if you already have one set up on your computer, it is fine to work with

How to Start Learning Python

My advice for beginners is to attempt it and not get occupied by small things. Just in case that you've never done anything related to machine learning, experiment with scikit-learn. You'll get a thought of how the cycle of labeling, preparing and testing works and how a model is created.

An alternative that you can experiment with machine learning begins with Keras —which is generally consented to be the most straightforward structure—and see where that takes you. After you have more involvement, you will begin to perceive your need from the structure: more prominent speed, an alternate API, or perhaps something different, and have the ability to settle on a more educated

choice.

It's essential to consider that every one of them has wide help and are enhancing continually, making correlations harder to make. A multi-month-old benchmark might be obsolete and year-old cases of structure X doesn't bolster activity Y could never again be substantial.

In case you're occupied with doing machine adapting particularly connected to NLP, take look at MonkeyLearn! Our stage gives an interesting UX that makes it effortless to manufacture, prepare, and enhance NLP models. You can either utilize pre-prepared models for normal utilize cases (like notion examination, subject location or watchword extraction) or prepare custom calculations utilizing your specific information. The basic framework or sending your models, our versatile cloud does this for you. You can begin at nothing and incorporate immediately with our excellent API.

Python is a mainstream and easy to use language. Python is an entire language and framework that can be used for both innovative works and create generational frameworks.

There are likewise a considerable number of modules and libraries to learn, giving a wide array of approaches to do each assignment. It can be overpowering.

The most ideal approach to begin utilizing Python for machine learning is to understand a few basic founding principles of the language.

It will compel you to introduce and begin the Python mediator.

It will give you a 10,000-foot perspective of how to advance through a little task.

It will give you certainty to go ahead to your own particular little tasks.

Beginners will need to start with a small project that goes from start to finish.

Books and even many available Python courses can be a bit perplexing. Most of them will give you loads of formulas and scraps; however, you rarely will get the chance to perceive how all of the pieces of the puzzle fit together.

When it comes time to apply what you have learned and learn about other particular Datasets, you are taking a shot at an opportunity for mental growth.

Some of the common tasks that make Python so simple to use are:

- Determine a Problem

- Gather the Data
- Analyze the Algorithms
- Increase positive results
- Define results & present them

The absolute easiest way to face with learning an unfamiliar platform or language is to never give up, and keep working through a machine learning case study until you get to the end. Just be sure to note all of the key points and techniques being taught along the way. In particular, pay special attention to the data loading, the data summarization, how to evaluate different algorithms, and how to come up with certain predictions.

If you are not able to do that, you have a template that you have available to you in each of the Datasets.

Try a new tool called the Iris Dataset; it is the easiest for those who are just starting out with learning machine language. This is a great project because of it's easy to use interface.

Attributes are generally numerical which means you have to figure out how to load and manage all of the data.

Due to its calculative nature, users are enabled to rehearse with a simpler kind of Python formula or code.

It is a multiple level arrangement issues (ostensible) that may require a detailed approach.

It just has four qualities and 150 columns which means it is extremely tiny and effectively fits into the memory

The majorities of the numeric qualities are in similar units and a similar scale and are not essential to any unique scaling or changes being made in order to begin.

We should begin with a simple program, "welcome world" machine learning program in Python.

During this segment, we will power through an easy and fun coding sample from start to finish.

Here are some of the common Python features we will discover during this section.

- Introducing the Python and i’s libraries
- Stacking the data set
- Condensing the data set
- Imagining the data set
- Assessing a few calculations
- Making a few forecasts

Take as much time as needed. Work through each progression.

Attempt to enter in these commands on your own, or try to reorder them for the sake of saving time.

Step 1: Download and Install

Get the Python and SciPy stage onto your computer by downloading it. This is assuming you do not have it on your machine already.

I would prefer not to cover in extraordinary detail since there are numerous articles on Google explaining how to do this.

Step 2: Instructional Exercise

This instructional exercise works with Python 2.7 or 3.5. have done some research on this topic and

Here is a list of libraries you need to have installed in order to complete this particular case study/exercise.

matplotlib

pandas

scipy

sklearn

numpy

Keep in mind there are numerous approaches to introduce Python libraries. My best guidance for newbies is to pick one technique at this beginning stage and maintain consistency.

I found that the SciPy page gives phenomenal guidelines for installation and introduction to the libraries on numerous distinctive platforms, for example, Linux, OS X, and of course, Windows. This guide is extremely popular,

thorough and widely used around the world.

In case you are using a Windows environment, or you are not certain, I highly suggest, that you install a clean and free version of Anaconda3. It is a free, open-source management system that will run on Windows, Linux, and even OS. It will help you in packaging and the distributing of your code.

Disclaimer: The example we are going to look at below is based on the assumption that you are at least using the 01.8 or higher version of Scikit-learn.

It is a smart thought to ensure that the Python environment was properly downloaded, set up properly, and it is functioning properly for this sample coding project.

The below coding script is going to help you to test out your working environment as well as automatically import each of the different libraries that are needed in this example.

Open a brand-new command line in your interpreter and type the following one word.

I definitely recommend working exclusively in the interpreter or composing your programs ahead of time, and then run all of them in the command line instead of using large editors and environments. Always keep things incredibly simple and concentrate on the code itself and not on the tools.

The interfaces will not change very fast; hence, don’t be alarmed if you happen to output an older version. All of this should still come out correct for you; however, if you do get any type of error, please don’t try to go on. You don’t want to learn to fix coding when just starting out. For some reason, you can’t execute the script without errors; the rest of the steps won’t work for you. There are plenty of forums for beginners on the web. Try searching your exact error code, in quotes, to see what the common fix is. Try the fix, if it’s easy enough for you to handle, then by all means, do so and come back to finish the rest of the steps.

The Basics of Working with Python

Before we get started with more of our machine learning, we need to understand some of the basic parts that come with the Python coding language. This is a great coding language to work with because it is simple and easy to use. Yet it still has a lot of power behind it. All the great features that come with Python, it is the perfect choice when working on machine learning.

Python Keywords

First, we need to take a look at these important keywords in the Python language. Like with what you will find in other coding languages there is a list of keywords in Python that are meant to tell your text editor what to do. These keywords are reserved, and you should only use them for their intended purposes if you want to be able to avoid issues with your code writing. They are basically the commands that will tell your compiler how to behave and they remain reserved so that you can execute the code without a lot of issues in the process.

Naming Identifiers in Your Code

When you are working with Python, it is important to follow the right rules when you are naming the different parts of your code. Several of these parts are known as identifiers. These can go by various names like variables, classes, entities, and functions. There are a few rules that you must follow when naming these identifiers, but the rules are going to be the same no matter which identifier you are working with. Some of the rules that identifiers need to follow when naming identifiers include:

- **You can use both uppercase and lowercase letters in the name of your identifier. You can also work with underscore symbols and numbers as well. Any combination of the above characters works as well, just make sure that inside the name, there aren't any spaces. So, do not write out something like 'My first program'. You would write it as 'Myfirstprogram'.**
- **Your identifier can't start out with a number. You can use the number anywhere else that you want in the name, but it can't be the first character of your program. If you do put a number as the first character, you are going to get an error signal when you have the compiler try to do this one. You can write something like 'one program', but you can't name the identifier something like '1program'.**
- **The name of the identifier can't have one of the Python keywords in it. If you add in the keyword to the name, you are going to cause confusion in the compiler, and you won't get the program to work.**

The rule for naming your identifier doesn't have to be difficult. If you do happen to forget one of the rules for naming an identifier, the compiler is going to

notice, and you will end up with a syntax error. You simply need to go back through and fix it and this error will go away.

Python Comments

You can also work with comments in Python. These comments can be useful for explaining some of your code, especially if you are trying to explain it to another programmer or person who is looking through your code. Any time that a party of your code needs a little bit of clarification, you can add in a comment. The compiler is not going to recognize the comment and simply skips right over it without reading at all. Once you indicate to the compiler that the comment is done, it will start reading the code again. Someone who executes the code will have no idea how many comments you have or even where you put them in the program.

It is pretty easy to write out the comments that you want to add into your code. You simply need to use the (#) sign in front of the comment that you want to write. So, you could write out something like #this is my Python code. When the comment is done, you just hit return and start writing the rest of the code on the next line. As long as you have that sign right in front of the comment, then your compiler is able to just skip over it and will not read out what you put in the comment.

You are able to choose how many comments you would like to do inside of your code. Sometimes the code will just need a few of them while other times the code may need a lot of comments. Keep the comments down to the ones that you really need and do not waste time or space writing out more comments which aren't necessary.

Python Statements

You can also work with statements inside your Python code. You can then send these statements over to your chosen compiler so that the code can be executed. You can write out any statement that you would like, but it must be written out in a manner that the compiler will understand. Statements can be as short or as long as you would like. Some statements are only going to be one character long, and other times it will be many lines long.

Python as an Object-Oriented Language

As you look through some of the different codes that we have for this kind of learning in the guidebook, you will notice that it is very much an Object-Oriented Programming or OOP language. This is a fairly new form of

programming, but you will find that it works way better for doing your programs and can make coding, especially with machine learning, easier for a beginner.

The OOP languages are basically designed to be easier and to cut out some of the issues that used to occur when programming. Some of the older coding languages are not designed this way, and that made it harder for a beginner to get started with.

OOP was designed in order to get rid of these issues because it will use the programming features that are best in a way that is more structured. Basically, this structure is going to help you as a programmer get the work done faster. If you tried out some of the older coding languages and got frustrated, you will find that Python got rid of some of these issues and coding is much easier to accomplish now than before.

Unlike Python, some of the earliest programming languages were released with a development method that is known as the procedural approach. This got the job done and did help the programmer to get their work done and it made some great codes for many years, however, there were a lot of flaws. These flaws were things that some programmers were able to get past, but they were often troublesome enough that beginners would get frustrated and walk away from doing the work at all.

Since the procedural approach was really hard to use in programming, the idea of OOP was developed. When you are using OOP, you will find that the data in your code is going to be treated as an important development of the code. In the procedural development, the data was allowed to just flow around in the system. OOP will do a much better job keeping the data in place so that the function that operates it will stay right next to it. This keeps all your data safe and makes sure that the programmer, or outside sources, aren't accidentally making modifications.

For the beginner who doesn't understand all of the technical things that come with programming, OOP is basically just going to provide you all of the power that you want when creating a new code. It makes things so much easier to use. There are a lot of features that come with OOP which helps to make it better to use including:

- **OOP is able to pull some of the emphasis away from the procedure of your code and it places more of this emphasis on the data in your code.**

- **All of the programs that you write will now be divided up into objects.**
- **The data is structured in order to be characterized by the objects that are inside of them. The data will be the classes that hold the objects, and all the objects that are inside of the class need to match up in a way that makes sense.**
- **The functions that will be important for operating the data of your objects are going to come together when you are working with the structure of the data.**
- **The data in your code is hidden so that the external functions will have access to the data. This sometimes caused some issues in the past when it came to the older programming languages because the information could easily get mixed up.**
- **The objects that you use have the ability to communicate with each other thanks to the fact that they have functions that are in common.**
- **The new data, as well as the functions that you have, can be added into the code any time that you want.**
- **Whenever you start to design one of your own programs, you have to remember that you have to follow the bottom-up approach.**

There is a lot to enjoy when using the Python coding language. It is designed with a beginner coder in mind, which is why it is one of the best options for you to choose and why you find it being used with some other complex topics such as machine learning and even hacking.

Chapter 8. Important python frameworks

Though it has been used to perform a variety of tasks ranging from data science and machine learning to robotics (especially AI) and hardware programming, web programming can also be done with Python. Nowadays, programmers give priority to the use of Python language in web programming to build applications more efficiently.

Some essential web frameworks are used to build applications. However, it is also a question for new programmers to understand which framework is better for their projects.

What is a web framework?

A web framework is actually the use of tools and resources to build and manage websites, web applications, and web services. The frameworks work as an abstraction tool that allows reusing the code and makes it easier to develop applications.

Why do we need web development frameworks?

Frameworks make it easier for a developer to develop software and websites.

The use of frameworks speed up the development, and hence, the development becomes faster, quicker, and more efficient. Most of the web frameworks are available at cheap costs. A framework has strong security implementations. The most benefits of using frameworks are actually due to the community behind them.

If you're browsing the introductory pages of this book on a preview or something of the like, on the fence as to whether or not you want to learn Python, well, I can tell you that you probably should. Well, Python is basically an incredibly useful and powerful language. It's present essentially everywhere. Everything from the scripting side of video games (or the video games themselves) to intensive server-side web applications to the plethora of deep and responsive desktop applications that have been built with it.

Tornado framework also helps in making thousands of concurrent connections to your application at the same time. Dash is also a web framework that is considered as one of the best choices for analytical applications.

Django:

Django is the most popular framework for web development that is simply defined as "the web framework with advanced features for perfectionists with deadlines." It has built-in features that are allowed for a wide range of web applications such as Chatbots, database applications, GPS solutions, etc.

Django has a philosophy that is called DRY (Do not Repeat Yourself) allows and promotes the reuse of code as well as slicing the code time in half. Its modular architecture allows modification of the code components by enabling the option to allow addition or removal of components as much as needed.

Furthermore, Django framework possesses ORM (Object Relational Mapping). The ORM makes it highly compatible with most of the popular databases, including SQL and Oracle. It has the ability to work with several databases at once.

Django is also an SEO friend framework by reducing the page loading time through different techniques and features like compressing JavaScript and caching templates.

Pyramid:

Pyramid is another framework for web applications which defines itself as "not too small, not too big, and just right." Pyramid has the ability to start small. It

allows a programmer to code a solid foundation for specific problem-solving projects and then to scale it up as required. In comparison with Django, Pyramid has the same compatibility with both small and large applications, but in complexity, it is not the same as Django.

Pyramid is an outshine framework with a plugin system which allows programmers to perform whatever is required. More specifically, these plugins help allows for the implementation of multiple solutions/results for a given problem.

Pyramid is ideal for single-file applications, authorization or apps oriented, and flexible authentication.

For server-side programming, Python has been widely used owing to its capabilities of dynamic website creation. Technology giants like NASA and Google use Python programming language due to its use in fast prototyping and the development of highly scalable web applications.

The Python frameworks make web development friction-less and reduce development efforts. The only challenge for programmers is to identify which framework they need for a specific task or better results.

The professional programmers, also called experts, have explained that according to the needs, benefits, uses, advanced features, and functionalities, programmers would most likely work with few frameworks in the long run. Here are the best frameworks of all time and also favorite for future use:

- CherryPy
- Tornado
- Bottle
- Flask
- CubicWeb

CherryPy:

CherryPy is one of the oldest Python frameworks but still with added functionalities and updated versions it is still most likely used framework. CherryPy is quite stable and fast as compared to other frameworks. Programmers can build web applications with the help of this framework in a similar way to how they build any other object-oriented application. It results in the development containing smaller source code in less time. Since its launch,

CherryPy has shown to be quick enough and efficient.

Features:

Following are the major features of CherryPy Python framework:

· CherryPy runs on Python of version 2.7+ and 3.5+, Android, PyPy, and Jython

· It has a flexible and specialized plugin system

· Its powerful configuration system is another feature for developers

· It is super simple and easy to run multiple HTTP servers at once

· CherryPy also offers the testing support, profiling support and built-in coverage in its features

· Its features also include built-in tools for encoding, caching, authentication, sessions, static content, etc.

Tornado:

Tornado is also one of the most popular Python frameworks. It has asynchronous networking library which can be used simultaneously in handling concurrent connections.

Features: Following are the listed major features of Tornado framework

· Its performance is of high quality

· Developers are allowed to read the source code anytime

· Since it is a small framework, it is best for HTTP+JSON services

· Non-blocking HTTP clients

Bottle:

Bottle is another web framework for the Python programming language. More specifically, it is a WSGI (Web Server Gateway Interface) micro-web framework and is designed in such a way that its use is super simple and its performance is super-fast and of lightweight (50-60 kb). Most interesting about this web framework that it has no dependencies except the Python standard library. It allows the development of web applications effectively and quickly.

Features: Bottle web framework has the following main features

· It helps in routing provided that a programmer must use simplified syntax or expressions for URL parameters

· This framework provides convenience to access form data, file uploads, headers, cookies, and other HTTP metadata

· It helps the developers to optimize the speed by testing the fast static routes and frequently used routes.

· It runs with both Python 2.5 and 3.5

Flask:

Features: Flask web framework has the following features

· It contains development server and debugger

· Extensive documentation can be done

· Provide support for secured cookies

· Provide integrated support for unit testing

· Unicode-based feature

CubicWeb:

CubicWeb is another web framework for Python, which is semantic, open-source web structure, and free. It encourages engineers to construct a web application productively with the help of reusing 3D shapes called segments and by following notable item situated plan standards. For the advancement of a semantic web application, its features are of great importance that includes quality, reusability, and productivity.

Features: CubicWeb has the following main features

· For common needs, a library of reusable components of data model and views is available

· It has a well-developed security workflow

· It supports OWL (Ontology Language) and RDF (Resource Description Framework).

Many web frameworks for Python have been discussed that give an idea to select and use the one that best matches the project requirements of a programmer. The increase in the usage and development of web applications

generally increase the use of such frameworks. Its use depends upon the business requirements and needs in order to decide which Python framework best matches your business. Both online and offline support is available on web services and in many different countries, respectively. These service providers can help programmers to choose the right framework for web development. However, it can also be done if programmers try to explore these web frameworks.

Registering Python Programs

The Python command line is an excellent environment, especially for writing and testing small pieces of code. However, as the code multiplies and grows, the command line will become insufficient. Moreover, of course you will want to save and store the code you write. This is where text editors come into play. You can use any text editor you want to write Python code. Not even a notepad. However, starting with a text editor that can distinguish and color Python code will make your life easier in every respect. As we said before, no matter which operating system you are using, I recommend that you read both Windows and GNU / Linux. Let's start with GNU / Linux: 3.1 GNU / Linux System If your system is a KDE desktop environment in GNU / Linux, any of the Kwrite or Kate text editors will suffice for the beginner level. At this stage, Kwrite may be recommended for its simplicity and simplicity of use. You can use Gedit if your system is the GNOME desktop environment in GNU / Linux.

Let's start by opening a new Kwrite document. The easiest way to open a new Kwrite document is by pressing ALT + F2 and typing: kwrite. To create a new Gedit document, we press ALT + F2 and give the command: gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 1 GNU / Linux System If the system you are using is a KDE desktop environment in GNU / Linux, any of the text editors named Kwrite or Kate will be sufficient for the beginner level. At this stage, Kwrite may be recommended for its simplicity and simplicity of use. You can use Gedit if your system is the GNOME desktop environment in GNU / Linux. Let's start by opening a new Kwrite document.

The easiest way to open a new Kwrite document is by pressing ALT + F2 and typing: kwrite. To create a new Gedit document, we press ALT + F2 and give the command: gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 1 GNU / Linux System If the system you are using is a KDE desktop environment in GNU / Linux, any of the text editors named Kwrite or Kate will be sufficient for the beginner level. At this stage, Kwrite may be recommended for its simplicity and simplicity of use. You can use Gedit if your

system is the GNOME desktop environment in GNU / Linux. Let's start by opening a new Kwrite document. The easiest way to open a new Kwrite document is by pressing ALT + F2 and typing: kwrite. To create a new Gedit document, we press ALT + F2 and give the command: gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 At this stage, Kwrite may be recommended for its simplicity and simplicity of use.

You can use Gedit if your system is the GNOME desktop environment in GNU / Linux. Let's start by opening a new Kwrite document. The easiest way to open a new Kwrite document is by pressing ALT + F2 and typing: kwrite. To create a new Gedit document, we press ALT + F2 and give the command: gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 At this stage, Kwrite may be recommended for its simplicity and simplicity of use. You can use Gedit if your system is the GNOME desktop environment in GNU / Linux. Let's start by opening a new Kwrite document. The easiest way to open a new Kwrite document is by pressing ALT + F2 and typing: kwrite. To create a new Gedit document, we press ALT + F2 and give the command: gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 We issue gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27 We issue gedit. After we get the empty Kwrite or Gedit document, the first line we need to do is: 27

#! / usr / bin / env will write python. So what does this command do? This command shows that the code that we will write next is a Python code and where Python is located on the system. So our system understands where to find Python. On GNU / Linux systems, Python's executable file is usually located in / usr / bin. So you can actually type the line above: #! / Usr / bin / python So your system will look for Python under the / usr / bin directory and understand that it should run the program you are writing with Python. However, on some GNU / Linux systems, Python's executable may be located in another directory (eg / usr / local / bin).

In this case, because the executable cannot be found under / usr / bin, under some circumstances the system will not be able to run the program you are writing. It may not be a good idea to give a fixed directory name as above, since the Python s executable file is not available in the same directory on every system. To avoid such system differences, we can use a script called env on GNU / Linux systems. This env script, which can be assumed to be located under / usr / bin, allows us to find out which directory Python is in. With the # / usr / bin / env python line we added at the beginning of our program, Python can

be easily detected wherever it is installed on the system. Long story short, when you type #! / Usr / bin / python, we give the system the following command: Search Python in / usr / bin! #! When you type / usr / bin / env python, you'll order: Find Python wherever it's hidden! Anyway ... Let's continue our way without further technical details.

In fact, writing code into text is not much different than writing code at the Python command line. Now let's insert the following lines into the Kwrite or Gedit document: #! / Usr / bin / env python a = "apple" b = "pear" c = "banana" print "a", a, "a", b, "a de ", c," I want to get "After you finish writing them, it came time to save our file. Now we save our file anywhere under the name experiment.py. Let's save our file on the desktop. We should now see a file called trial.py on the desktop. Now we open a console screen immediately. (But we don't run the Python command line.) So we come to where we saved the file: cd Desktop Of course here it’s assumed you were in your home directory. The first time you start the command line, your current directory will be your home directory (/ home / user_name). Therefore, the cd Desktop command shown above takes you to the directory where the desktop is located. Of course, if you open the command line in a different directory, the command above alone will not take you to the desktop. If so, you should issue a command similar to this: Of course, if you open the command line in a different directory, the command above alone will not take you to the desktop. If so, you should issue a command similar to this: Of course, if you open the command line in a different directory, the command above alone will not take you to the desktop. If so, you should issue a command similar to this:

cd / home / user_name / desktop

Note: On some Turkish GNU / Linux systems, the directory that displays the desktop may be called Desktop instead of Desktop. If so, you should of course replace the word Desktop with the word Desktop. If you've come to your desktop successfully, run the program you're writing, then press ENTER: It can save and run the file using the text editor named Gedit instead of Kwrite. As you can see, we can run our programs with the command python trial.py. Under normal circumstances, this is the preferred way of running the program. Well, but is there a way to run Python programs without adding a python command to the top? Here comes the importance of the #! / Usr / bin / env python line we just mentioned. In order to run our program without introducing a python command, we first need to authorize it. We do this with the help of the following command:

First: cd Desktop command where we save the file to the desktop. After that:

chmod a + x test.py command, we give the permission to run the file named test.py, that is, we make the file executable. Now we can run our program by running the following command on the command line: ./ experiment.py So, what does all of this have to do with the #! / Usr / bin / env python line? If we don't put this line in the text, The. / test.py command will not work because the system cannot understand which program to run this script. When we run python test.py, it is understood that the Python program will run the script. However, when we remove the leading python statement, if #! / Usr / bin / env python does not exist in the first line of the script, the system cannot decide what to do with it. If you type this line as #! / Usr / bin / python and if the Python executable file is not in the / usr / bin / directory on the system where your program is running, you will get an error output like: bash: ./ trial.py: / usr / bin / python: bad interpreter: No such file or directory In order to avoid such an error, we try to write that line in the form of #! / usr / bin / env python. So no matter where Python is installed, our system can detect Python. The py command will not work. When we run python test.py, it is understood that the Python program will run the script.

However, when we remove the leading python statement, if #! / Usr / bin / env python does not exist in the first line of the script, the system cannot decide what to do with it. If you type this line as #! / Usr / bin / python and if the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error output like: bash: ./ trial.py: / usr / bin / python: bad interpreter: No such file or directory In order to avoid such an error, we try to write that line as #! / usr / bin / env python.

So no matter where Python is installed, our system can detect Python. The py command will not work. When we run python test.py, it is understood that the Python program will run the script. However, when we remove the leading python statement, if #! / Usr / bin / env python does not exist in the first line of the script, the system cannot decide what to do with it. If you type this line as #! / Usr / bin / python and if the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error output like: bash: ./ trial.py: / usr / bin / python: bad interpreter: No such file or directory In order to avoid such an error, we try to write that line as #! / usr / bin / env python. So no matter where Python is installed, our system can detect Python. However, when we remove the leading python statement, if #! / Usr / bin / env python does not exist in the first line of the script, the system cannot decide what to do with it. If you type this line as #! / Usr / bin / python and if the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error output like: bash: ./ trial.py: / usr / bin /

python: bad interpreter: No such file or directory In order to avoid such an error, we try to write that line as #! / usr / bin / env python. So no matter where Python is installed, our system can detect Python.

However, when we remove the leading python statement, if #! / Usr / bin / env python does not exist in the first line of the script, the system cannot decide what to do with it. If you type this line as #! / Usr / bin / python and if the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error output like: bash: ./ trial.py: / usr / bin / python: bad interpreter: No such file or directory In order to avoid such an error, we try to write that line as #! / usr / bin / env python.

So no matter where Python is installed, our system can detect Python. If you type / usr / bin / python and the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error similar to this: bash: ./ trial.py: / usr / bin / python: bad interpreter: In order to avoid such an error, we try to write that line as #! / Usr / bin / env python. So no matter where Python is installed, our system can detect Python. If you type / usr / bin / python and the Python executable is not in the / usr / bin / directory on the system where your program is running, you will get an error similar to this: bash: ./ trial.py: / usr / bin / python: bad interpreter: In order to avoid such an error, we try to write that line as #! / Usr / bin / env python. So no matter where Python is installed, our system can detect Python.

After the above operations, we can remove the extension at the end of the name of this experiment.py file and run our program with the. What if we want to run our program just by typing its name? In order to do this, our program must be included in the PATH variable that is to say in Turkish, the program must be on the PATH. What does it mean when a program is on the WAY? As is known, the path to a program or file is roughly the directory in which that program or file is located. For example, the path to the fstab file is / etc / fstab. To give another example, the path to the xorg.conf file is /etc/x11/xorg.conf. This road also has a more specific meaning.

In the computer language, the executable files (.bin and .bin). sh files are executable files) in which the paths are specifically named PATH, and when used in this sense, the path word is usually capitalized. When we call a program, the operating systems look in some special directories to run the program. If the executable files are in these special directories, the operating system finds them and makes them run. So we can call these programs directly by name.

Now let's do a few experiments to understand this issue better. We immediately open a console screen and issue the following command: echo $ PATH The output of this command would look something like this: / usr / local / sbin: / usr / local / bin: : / usr / games This output shows what we call the PATH variable. Our operating system, when we want to call a program, it will first check the directories named above. If the executable file of the program we are calling is in any of these directories, we can call that program by name. For example; Let's take a look at the output of which gedit: / usr / bin / gedit As you can see, the path to Gedit is / usr / bin / gedit. Just above, we check the output of the echo $ PATH command and see that the / usr / bin / directory exists in the PATH variables. Therefore, Gedit program is on the WAY, we say. With Gedit already on the PATH, we can run Gedit by simply giving gedit to the console. Now, if we want to call the program directly by name, we need to copy it into one of the directories given in the echo $ PATH output. For example, let's copy our program into / usr / bin. Of course, you must have root privileges to copy a file into this directory. The following command will work: sudo cp Desktop / trial / usr / bin Now on the console screen:

When we write an experiment we need to see that our program works. Rather than adding our program file to one of the directories on the PATH, it is also possible to add the path to the path to our PATH. So: PATH = $ PATH: $ HOME / Desktop This is how we added the desktop directory to the PATH. If you want; With the echo $ PATH command, you can check if your desktop appears on the PATH. This way, executable files on the desktop can now be called with their own names.

However, when you add the desktop to the PATH, your desktop will not always remain on the PATH, and everything will be restored when the current console session is closed. If you want to permanently add any directory to the PATH, you must make changes to your .profile file. For example, to add the desktop to PATH, you must add the following line to the end of your /home/user_name/.profile: export PATH = $ PATH: $ HOME / Desktop Are you tired? Or are you bored? I think neither tired, nor bored ... Because the information we provide in this section, Python with a program to write this program, you use the distribution you use KDE or GNOME menu will be very useful for you to add. If you want to do any programming under GNU / Linux, you need to assimilate the above information.

Windows users ... 3.2 Windows System Windows users can work with IDLE text editor. To access IDLE, go to Start / Programs / Python / IDLE (Python

GUI). When IDLE is first started, it will look like this: it is very useful if you want to add the distribution to your KDE or GNOME menu. If you want to do any programming under GNU / Linux, you need to assimilate the above information. Windows users ... 3.2 Windows System Windows users can work with IDLE text editor. To access IDLE, go to Start / Programs / Python / IDLE (Python GUI). When IDLE is first started, it will look like this: it is very useful if you want to add the distribution to your KDE or GNOME menu. If you want to do any programming under GNU / Linux, you need to assimilate the above information. Windows users ... 3.2 Windows System Windows users can work with IDLE text editor. To access IDLE, go to Start / Programs / Python / IDLE (Python GUI). When IDLE is first started, it will look like this: To access IDLE, go to Start / Programs / Python / IDLE (Python GUI). When IDLE is first started, it will look like this: To access IDLE, go to Start / Programs / Python / IDLE (Python GUI). When IDLE is first started, it will look like this:

As we can see from the >>> sign that we see here, this screen is actually the Python interactive shell itself ... If you wish, you can do the operations in the interactive shell on this screen. But right now, our goal is not to play with the interactive shell. We are looking for a text editor that can write our Python programs. Clicking on the New Window button in the File menu brings up the following screen:

This is where we write the Python code. Now we add the following codes to this blank page: a = "apple" b = "pear" c = "banana" print "a", a, "a", b, "a", c, "I want to get" After writing our codes all we need to do is save the file somewhere. To do this, follow the path File / Save as and save the file to the desktop as trial.py. After saving the file, we can run the program we have written by following the path of Run / Run Module or directly by pressing F5 key. In this case we get an image like this:

If you want to run your program, do the following: 1. Follow the Start / Run path, type cmd in the pop-up window, and press ENTER. 2. Navigate to the desktop, where you saved the file by giving the following command: cd Desktop The first time the command line is opened in Windows, you will be in a directory such as C: \ Documents and Settings \ user_name.

Therefore, the cd Desktop command shown above will take you to the directory where the desktop is located. Of course, if you open the command line in a different directory, the command above alone will not take you to the desktop. In this case, you should issue a command similar to this: cd C: / Documents and

Settings / User_name / Desktop After coming to the desktop, you can run your program by running python. py Of course I've assumed that you've added Python to the PATH in the same way we explained here. If you haven't added Python to PATH, you can also use the following method to run your program: c: / python26 / python trial.py If everything went well, we need to get the following output: an apple, a pear, and a banana. In the meantime, we've been talking about ROAD a few times.

In the last chapter we even learned how to add Python to the PATH. But what is this ROAD? Let's take a little look at this. If you want to write a Python program and present it to your users in the future: py If everything went well, we need to get the following output: I want to buy an apple, a pear and a banana. In the meantime, we've been talking about ROAD a few times.

The path to a program or file is roughly the directory in which that program or file is located. For example, the path to Internet Explorer is C: \ Program Files \ Internet Explorer \ iexplore.exe. So you can start IE from the command line with the following command: "C: \ Program Files \ Internet Explorer \ iexplore.exe" To give another example, the path to the hosts file is C: \ WINDOWS \ system32 \ drivers \ etc \ hosts. So you can issue the following command to view the contents of this file: notepad "C: \ WINDOWS \ system32 \ drivers \ etc \ hosts" The path to Python is C: \ Python26 \ python.exe. So we can issue the following command to run Python: C: \ Python26 \ python.exe This path word also has a more specific meaning. In the computer language, executable files (. exe files are executable files) and the paths in which they are used are referred to as paths.

When we call a program, the operating systems look in some special directories to run the program. If the executable files are in these special directories, the operating system finds them and makes them run. So we can call these programs directly by name. Now let's do a few experiments to better understand this issue. Let's start MS-DOS followed by Start / Run, command cmd in the pop-up window and write a command like this: echo% PATH% The output of this command will look something like this: C: \ WINDOWS \ system32; C: \ WINDOWS; C : \ WINDOWS \ system32 \ Wbem; This output shows what we call the PATH variable in English. When we want to call a program, our operating system will first check inside the directories named above. If the executable file of the program we are calling is in any of these directories, we can call that program by name.

Note: Note that the PATH directories are separated by semicolons! Therefore, if you want to add a new directory to the PATH, you must separate it from the

previous directory with a semicolon. For example, notepad.exe is a program located under the C: \ Windows directory. Since this directory is located on the PATH, we can call Notepad directly by name. You can also test it yourself by giving the notepad command on the command line. In the same way calc.exe C: The program is located under the \ Windows \ System32 folder. Since this folder is on the PATH, we can call the Calc program directly by name ... Now if we want to call the program directly by name, we need to copy the program into one of the directories given in echo% PATH% output. Let's copy our program into C: \ Windows directory. Now on the command line: typing experiment.py, we should see that our program is running.

Rather than adding our program file to one of the directories on the PATH, it is also possible to add the path to the path to our PATH. To do this, we need to issue a command in MS-DOS as follows: PATH =% PATH%;% USERPROFILE% \ Desktop If you want; With echo% PATH% you can check if your desktop appears on the PATH. This way, executable files on the desktop can now be called with their own names. However, when you add the desktop to the PATH, your desktop will not always remain on the PATH, and everything will be restored when the current MS-DOS session is closed. If you want to permanently add the desktop directory to the PATH, you must write it to the PATH list in System Variables, as we have shown before.

Here comes the question immediately: We were able to call programs such as Notepad and Calc directly with their names without specifying the extension.exe. But for example, to run our program named trial.py, we had to specify the extension. What is the reason for this? Windows has a special variable called PATHEXT. This variable holds extensions that the system considers executable. Echo% PATHEXT% Here we get the following output: .com; .exe; .bat; .cmd; .vbs; .vbe; .js; .jse; .wsf; .wsh if it has one, it will be considered executable. As you can see, .exe is one of these extensions. Therefore, we can call a file with this extension without specifying its extension. You can also add the extension to this list if you want. To do this, you can:

1. Double-click the System icon in the Control Panel.

2. Enter the Advanced tab and press the Environment Variables button.

3. Locate and double-click PATHEXT in the System Variables section.

4. After the last entry, w.wsh, add the entry .PY.

5. Press OK to exit. So now you can also run files with the .p extension without

specifying the extension ... Python registers itself in Windows registry. So Windows knows how to run Python programs. In fact, this way we write Python programs on Windows, it is also possible to run it using only the name in the directory where the program is located. So if we assume that we saved the program on the desktop and we think that we came to the desktop with the following command: cd C: / Documents and Settings / User_name / Desktop After coming to the desktop by giving the following command we have the opportunity to run your program:

However, for this command to work, we need to be in MS-DOS in the current directory. This means that if your program is in the My Documents directory and you are in the C: / Documents and Settings / User_name / Desktop directory in MS-DOS, this command will not work. To call your program from anywhere, no matter where you are, you should add your program to the WAY as we just explained ... Anyway ... Let's go back to our topic now. What were we saying? Yes, can't we double-click the programs we've written? Of course we can.

Since Python registers itself when installing on Windows, our program will run when we double click on the file with the extension. But there is a problem here. When we double-click our file to run our program, but this process is so fast that we only see that a black command line is opened and closed at the last speed ... In fact, our program is running, but because it runs so quickly, we cannot see that our program is running.

To see that our program is running, we need to prevent that black command line from closing. We do this by adding the following line to our program: raw_input () So our code should look like this: a = "apple" b = "pear" c = "banana" print "a", a, "a", b, "a in ", c," I want to get "raw_input () After a few sections, we'll examine what this raw_input () statement we added to the last line is and exactly what it does. For now, just know that this line is preventing MS-DOS from shutting down. With this command, MS-DOS will remain open until you press ENTER ... But at this point we need to make a very important warning. When developing a program with Python, never use this double-click method to run your programs. Always run your programs from the command line. So, if there is an error in your program, you can understand where this error originated. But if you run your program by double-clicking, you won't have the chance to see the error. Let us give an example.

The field formula of a triangle is: area = base x (height / 2) Our codes should be: height = 4 base = 6 area = base * (height / 2) print area Here's what we do:

First we define a variable called height. The value of this variable is 4 But at this point, we need to make a very important warning. When developing a program with Python, never use this double-click method to run your programs. Always run your programs from the command line. So, if there is an error in your program, you can understand where this error originated. But if you run your program by double-clicking, you won't have the chance to see the error.

Then we have defined another variable named base. The value of this variable is 6 3. We write our formula that allows us to calculate the area of the triangle: field = base * (height / 2) 4. Finally, we print the value of the variable named field with the help of the print command. 5. When we run this program, the result will be 12 ... Now think about it. Let's say we made a small mistake while writing this program and we came up with something like this: height = 4 base = 6 ala = base * (height / 2) print field Because the MS-DOS screen will turn on and off at the last speed. Moreover, adding raw_input () to the last line is useless. This line allows only error-free programs to be run by double-clicking.

To avoid such an error message, we need to add the following line between our codes: # - * - coding: utf-8 - * - Windows users may prefer to use character encoding cp1254 instead of utf-8: # - * - coding: cp1254 - * - In this way, we introduce the character type we use to Python. Ascii ahead, We will examine why Python treats Turkish characters like this when handling unicode and Python. But don't worry too much about this for now. The latest version of our program will look like this: # - * - coding: utf-8 - * - centigrade = 22 fahrenheit = centigrade * (9.0 / 5.0) + 32 print "% s corresponds to degrees s degrees fahrenheit \". % (centigrade, fahrenheit) As you can see, this time our program was able to show Turkish characters properly. In the meantime, let's take a look at the codes above:

Then we have defined a variable called Celsius and set the value to 22. Here, we first wrote # - * - coding: utf-8 - * - Thanks to this line, we introduced the encoding format that we will use to Python. Thus, we will be able to use Turkish characters in our codes. This line of Windows ta - * - coding: cp1254 - * -. Then we wrote our formula to convert the fahrenheit to centigrade. Note that the formula says 9.0 / 5.0 instead of 9/5. Thus, we were able to divide the number 9 precisely by the number 5 otherwise, our formula will calculate the result incorrectly. Finally, we showed the user how many degrees Fahrenheit corresponds to 22 degrees Celsius. Notice how we format the string here. When Windows users running IDLE try to register this program without typing # - * - coding: cp1254 - * -, they will see a warning window like this: Thus, we were

able to divide the number 9 precisely by the number

Chapter 9. Introduction to data analysis

Data analysis includes all the techniques and processes used in extracting information from raw data. Since raw data is usually unstructured in form, and hardly informative, the need to organize such data becomes even more important. While there are many extra tools that can be used for handling data analysis i.e. Microsoft Excel, R-language, SQL, etc., most data scientist prefer to use Python due to its extensive libraries and support packages for data analysis. The most popular packages/frameworks which we would be exploring in this chapter are NumPy and Pandas.

To risk over-simplification, data science is just the application of various techniques – usually employing the fast and efficient data organization, visualization and interpretation of computer programs or software to transform raw data into information for decision-making. Advertisements are better targeted at consumers, companies are more aware of their economic performance and possible trends or options in production, etc. If you're browsing the introductory pages of this book on a preview or something of the like, on the fence as to whether or not you want to learn Python, well, I can tell you that you probably should.

As you continue to grow as a programmer after this book, you're going to learn when you should and shouldn't use Python just as a matter of intuition. Python is an absolutely fantastic language, but the place where it fails is when you have to get extremely close to a computer's hardware or write incredibly efficient programs. In these areas, Python doesn't excel.

However, that's not to say it doesn't have its perks. In fact, I'd say that's one of the few places that Python falls flat. And what it lacks there, it makes up for in other areas. For example, development time in Python is generally extremely low in comparison to other languages.

However, without the right tools, guidance, and dedication, mastering the skills required for data science would be a very tedious and lengthy task. This is the reason for writing this book; to bring you up to speed on the skills and tools to begin a journey into the exciting and rewarding world of data science.

At this point, you are probably quite excited about data science and the secrets it has to offer, however, you may ask "what does Python have to do with it?"

Remember, data science leverages the exceptional processing and data manipulation capacity of computers? To do this, the data scientist must communicate the task in a clear and logical manner to the computer.

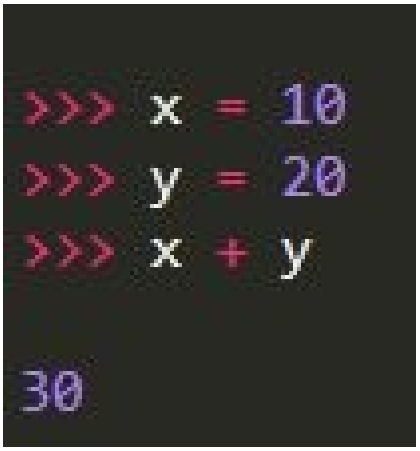
Also, Python has an extensive library of tools and applications that makes data science less technical for beginners or intermediate programmers. Therefore, most data scientists have Python as their primary tool for big data, data analysis, etc.

Chapter 10. Algorithms for starters

First, we shall start to look at Python operators:

Python operators can be described as special symbols that denote a specific operation. The values that the operator works on are referred to as operands.

See this example:



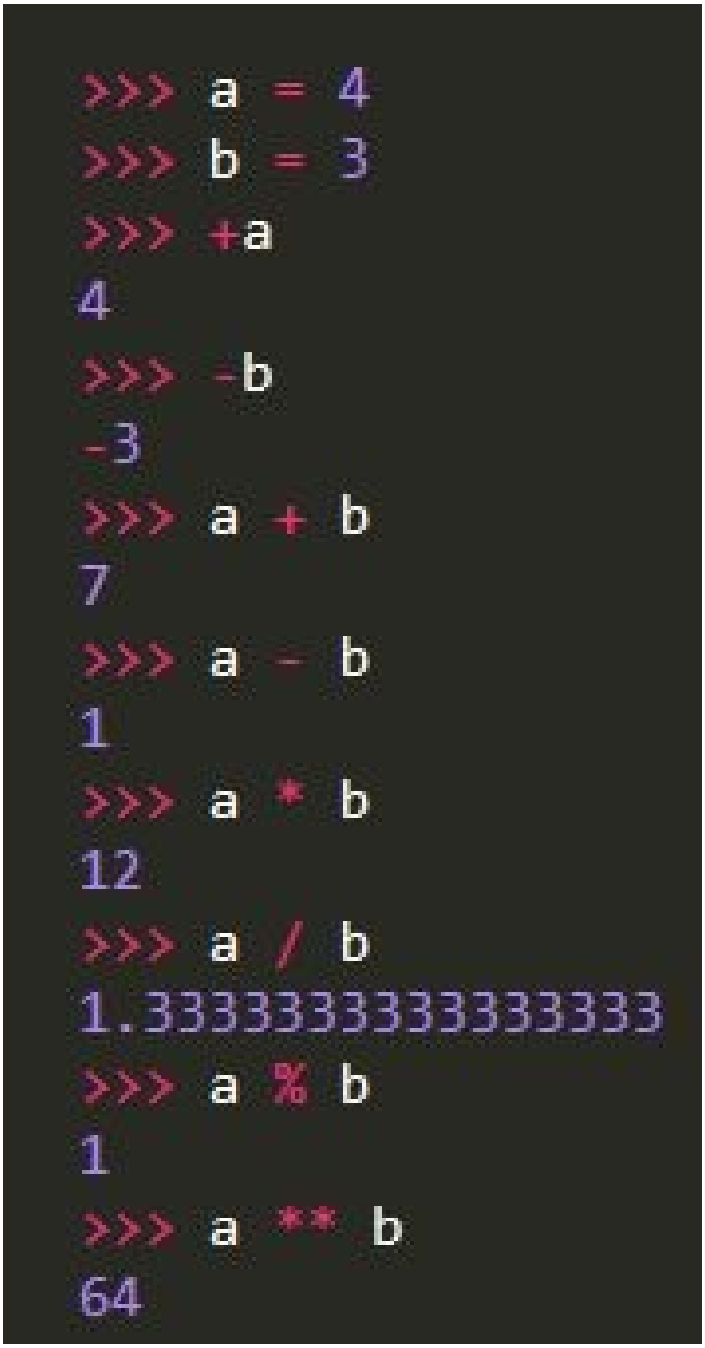
In this example, the + operator computes the sum of x and y. An operand can be a literal value or even a variable which references an object.

When you have a series of operators like x + y – z, this is called an expression. Python allows many operators to combine data objects into expressions. These are shown here:

Arithmetic Operators

Arithmetic operators include +, -, **, //, %, and /.

An example to show some of these operators in a program include:



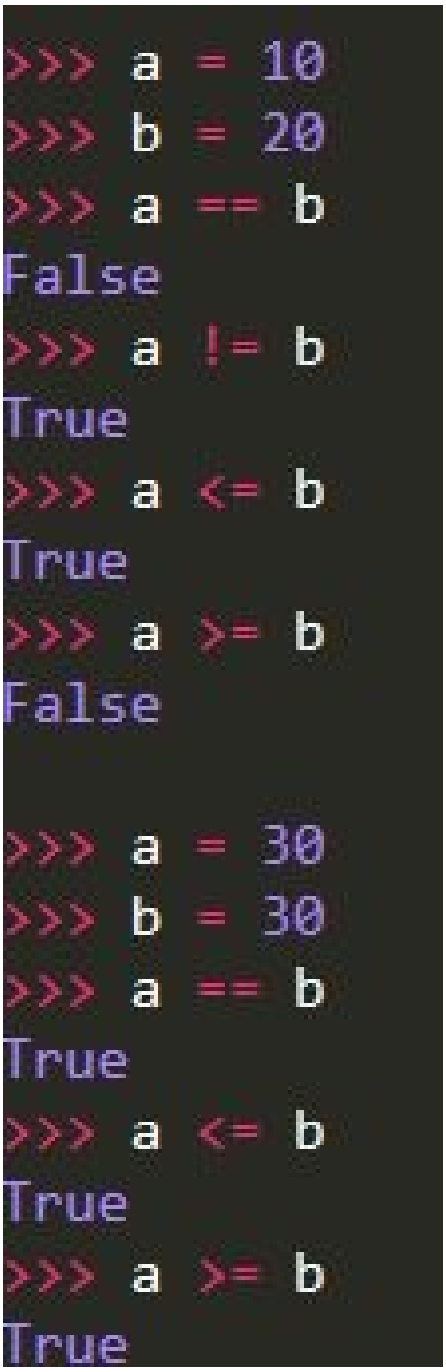
The output of a standard division (/) is a float, even when the dividend is evenly divisible by the divisor.

If the result of a floor division (//) is positive, it is like the fractional side is truncated off, leaving the integer part. When the result is negative, it has to be rounded down to the next smallest integer:

```
>>> a = 4
>>> b = 3
>>> +a
4
>>> -b
-3
>>> a + b
7
>>> a - b
1
>>> a * b
12
>>> a / b
1.3333333333333333
>>> a % b
1
>>> a ** b
64
```

Comparison Operators

These operators include ==, <=, >,!=, < and >=. Examples of comparison operators include:



Typically, comparison operators are commonly used in Boolean contexts such as conditional and loop statements to guide program flow.

Equality Comparison-Floating-Point Values

The values of a float object stored internally may not be accurately what you think it can be. As a result, it is a waste of time to compare floating-point values for exact equality. Take, for example:

```
>>> x = 1.1 + 2.2
>>> x == 3.3
False
```

The inner representation of addition operands doesn't reflect exactly to 1.1 and 2.2, and thus you cannot depend on x to make a comparison to 3.3.

The right way to check if two floating-point values are "equal" is to determine if they are close to each other. For example:

```
>>> tolerance = 0.00001
>>> x = 1.1 + 2.2
>>> abs(x - 3.3) < tolerance
True
```

abs () displays an absolute value. If the difference in absolute value between the two numbers is less than a certain tolerance, they are near to be said to be equal.

Logical Operators

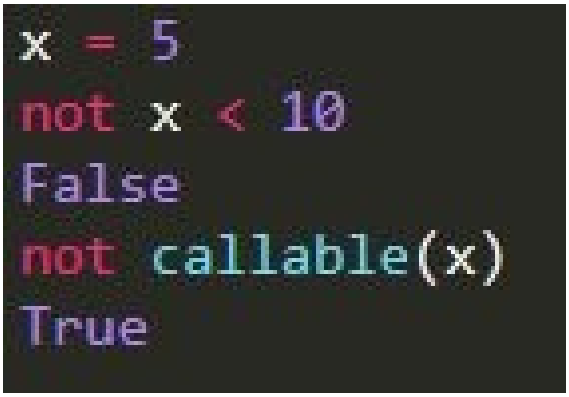
The logical operators like or, not, and, "and" change and combine expressions computed in Boolean context to calculate more complex operations.

Logical Expressions including Boolean Operands

There are specific objects and expressions within Python that are of Boolean type. In other words, they are equivalent to one of Python objects False or True. Take, for example:

Explanation of logical expressions that include or, not, and "and" is simple when the operands are Boolean.

See how they work in a program:



Operand	Value	Logical Expression	**Value**
x < 10	True	not x < 10	**False**
Callable (x)	False	not callable (x)	**True**

Computation of Non-Boolean Values in a Boolean Context

Most expressions and objects aren't equal to True or False. Even so, they cannot be determined in a Boolean context to be "falsy" or "truthy."

Well, what is true and what isn't?

In python, this is considered false when computed in Boolean context:

- **The Boolean value False.**
- **An empty string.**
- **The unique value represented by the Python None keyword.**
- **Any value that is numerically zero.**

Nearly all other objects defined in Python is considered as true.

You can tell the "truthiness" of an expression or object using the built-in bool () function. The bool () returns True if the argument is true, and False when it is Falsy.

The Compound “Or” Expressions

Take a look at this expression:

```
x1 or x2 or x3 or … xn
```

This expression is true if one of the xi is true.

In this expression, Python implements a methodology known as short-circuit evaluation. The xi operands are computed in order from left to right. Once it is verified to be true, the whole expression is considered to be true. At this stage, Python stops and no more terms are determined. The value of the whole expression is the xi that completed the evaluation.

For example:

```
>>> f(0)
-> f(0) = 0
0

>>> f(False)
-> f(False) = False
False

>>> f(1.5)
-> f(1.5) = 1.5
1.5
```

Since f () will return the argument passed to it, it can make the expression truthy or falsy by defining the value of arg that is truthy or falsy. Also, f () outputs its argument to the console, which proves whether or not it was called.

Now, take a look at the following compound logical expression:

```
>>> f(0) or f(False) or f(1) or f(2) or f(3)
-> f(0) = 0
-> f(False) = False
-> f(1) = 1
1
```

The Python interpreter will first determine f (0); the result is 0. A numeric value of 0 is false. This expression isn’t true yet, and thus, the computation starts from left to right. The next operand, f (False), will return False. This is also false, and the evaluation continues.

Chapter 11. Conclusion

First, Python is open source, cross platform and nearly ubiquitous. In fact, if you are using a Mac or Linux system there is a good chance it is already installed on your computer. To check, open up a terminal window and issue the following command:

python –V

If python is installed you should get a response like:

Python 2.6.1

Which tells you the current version of Python installed on your system? If it is not installed, a quick trip to http://www.python.org/downloads will give you download access to Python installers for a broad range of operating systems.

For advanced users, the Python source code is also available from that location which allows for the modification and/or custom compiling of Python to suit specific applications. Aside from having Python installed, all that is needed to start programming is a text editor. The process can be streamlined and made easier with a Python friendly IDE, but it is not required.

Next, Python is incredibly easy to use and extremely powerful. While Python code can be compiled and packaged into an executable, it is typically used as an interpreted scripting language. As such, housekeeping chores such as memory management are handled by the interpreter, freeing the programmer to focus on function.

Additionally, the interpreter is compiled for the operating system it is installed on so Python programs can be fairly system independent. In other words, code written on a Mac will in many cases run fine on a Windows computer, or even a Raspberry Pi.

Python also uses very simple 'readable' syntax which is similar in format to other languages. Someone new to programming can be creating simple programs within a few minutes. Those with programming experience in other modern languages (like C/C++) will find the syntax familiar, facilitating a quick jump into more complex tasks.

Finally, all the benefits offered by Python have made it one of the most popular programming languages in use today. That broad user base coupled with its open source nature translates to a huge, easily accessible resource pool. Resources from application specific libraries to online tutorials, online forums, and even college courses are never far away. This level of community support is invaluable to users from the novice level to top tier pros.

Whether your task is to sort incoming emails on a server, or build a control system for a revolutionary deep sea drone, Python provides an easy to learn open source solution with unparalleled community support to get the job done quickly and efficiently.